Universidade Federal Rural de Pernambuco
Departamento de Informática
Curso: Bacharelado em Sistemas de Informação

Elizangela de Lima Lucena

ANÁLISE COMPARATIVA ENTRE FRAMEWORKS DE INTERFACE RICA PARA JAVASERVER FACES

Recife
Julho de 2014

ELIZANGELA DE LIMA LUCENA

ANÁLISE COMPARATIVA ENTRE FRAMEWORKS DE INTERFACE RICA PARA JAVASERVER FACES

Monografia apresentada ao curso de Sistemas de Informação, como requisito parcial para obtenção do título de bacharel em Sistemas de informação.

Orientador: Prof. Wylliams Barbosa

Recife 2014

A meu querido esposo, Julio Cezar

Agradecimentos

A Deus.

Ao professor Wylliams Barbosa por ter aceitado ser meu orientador e pelas importantes colocações que direcionaram o rumo dessa pesquisa.

Aos professores Jones Albuquerque e Silvana Bocanegra pelo eterno incentivo e dedicação devotada aos alunos do nosso curso.

A minha família, em especial a minha mãe que sempre me encoraja e estimula a estudar e alcançar meus objetivos. Por fim, mas não menos importante, a meu esposo, pela paciência nos momentos de stress nos momentos de ausência presente.

Resumo

Este projeto tem como objetivo propor a apresentação de forma comparativa de três frameworks de componentes de interface rica para JavaServer Faces (JSF): Richfaces, Icefaces e Primefaces. Nosso intuito é criar uma ferramenta que automatize o processo de comparação e direcione os profissionais para a adoção do framework que melhor se adeque as suas necessidades. Para isso o trabalho apresenta uma breve introdução conceitual sobre o JSF, em seguida serão apresentadas as suítes de componentes Richfaces, Icefaces e Primefaces. Daí, explicamos os critérios, os métodos de avaliação e análise que foram utilizados para faze a comparação. Em seguida, detalharemos a criação da ferramenta que automatiza o processo comparativo e por fim seu uso de forma experimental em um caso real.

Abstract

This project proposes to present comparative form of three frameworks rich interface components for JavaServer Faces (JSF): Richfaces, Icefaces and Primefaces. Our aim is to create a tool that automates the comparison process and direct professionals to adoptiong the framework that best suits their needs. We present a brief introduction on conceptual JSF then the suites Richfaces, Icefaces and Primefaces components will be presented. Hence, we discuss the comparison explaining the criteria, assessment methods and analysis that were used. Then we detail the creation of the tool that automates the comparison and finally process their use experimentally in a real case.

Sumário

Lista de tabelas

Lista de figuras

Capítulo 1

Este capítulo apresenta o contexto no qual se insere o trabalho, sua justificativa e seus objetivos (gerais e específicos).

Introdução

1.1 Contextualização

A impressionante marca de 2,4 bilhões de internautas divulgada pelo site Pingdom em 2012 nos mostra a dimensão da web. Ainda segundo o site, existem 634 milhões de sites na web, 200 milhões de pessoas ativas no twitter mensalmente e 1 bilhão de pessoas ativas no facebook [37].

O crescente número de usuários da Internet fez surgir inúmeras redes sociais, sites para comércio e aplicações de diversos tipos na Internet, nessas aplicações nos deparamos com facilidade de uso, interatividade e riqueza estética. São esses atributos que alimentam as expectativas dos usuários quando estão no ambiente web. Com o avanço dos serviços ofertados e da exigência de seus usuários surgiu à necessidade de criar aplicações cada vez mais complexas que se aproximem em aparência, comportamento, usabilidade e segurança das aplicações desktop [7].

No intuito de tornar mais ágil o desenvolvimento das aplicações web, reduzindo a complexidade e aumentando a produtividade surgiu o Java Server Faces (JSF). Framework baseado no padrão Model-View-Controller (MVC) que torna o desenvolvimento de sistemas menos complicado devido à separação entre a interface e as regras de negócio. Para o desenvolvimento de interfaces o JSF fornece um conjunto básico de componentes através de suas duas bibliotecas

“HTML” e “Core”. Além dessas bibliotecas bases de componentes, o JSF suporta a extensão e criação de novos componentes através de interfaces ricas que tem seus fundamentos baseados na tecnologia Rich Internet Applcation (RIA) [1].

As aplicações RIA têm como objetivo trazer para o ambiente web características e funcionalidades que anteriormente eram exclusivas de aplicações desktop. As vantagens de se usar RIA não se limitam apenas à interface visual, existem outros benefícios como possibilidade de executar processamento assíncrono, interface mais reativa com resposta mais rápida para o usuário, equilíbrio de processamento entre o cliente e o servidor e otimização da rede com significativa redução do fluxo de dados porque muitos dados só são transferidos quando é realmente necessário [61].

Existe uma ampla variedade de bibliotecas de interfaces ricas para JSF atualmente no mercado, esse cenário pode deixar confuso o desenvolvedor no momento de escolher suas ferramentas de trabalho. Essa situação é agravada pela falta de parâmetros claros sobre qual opção é a mais indicada para determinado tipo de projeto.

1.2 Justificativa

O número reduzido de estudos onde resultados positivos direcionem de forma técnica a adoção de suítes de componente JSF determinando quando cada suíte é indicada.

1.3 Objetivos

O objetivo do trabalho é apresentar uma análise comparativa entre suítes de componentes JSF com a finalidade de gerar uma ferramenta que automatize esse processo de comparação. Em seguida aplicá-la durante a construção de uma software em um problema real. Demonstrar as vantagens e os benefícios obtidos.

1.4 A Organização do documento

Este trabalho está organizado em capítulos, dos quais este é o primeiro e apresenta o contexto ao qual o trabalho está inserido, a justificativa e o objetivo. O Capítulo 2 apresenta o referencial teórico que é baseado em conceitos de aplicações para web,

frameworks especificamente JSF, biblioteca de componentes para JSF e métodos comparativos. No Capitulo 3 temos a origem, características e funcionamento do JSF enquanto o capítulo 4 apresenta as bibliotecas de componentes Richfaces, Icefaces e Primefaces. A análise comparativa é apresentada no capítulo 5. Em seguida temos o capítulo 6 com a elaboração da ferramenta que automatiza a análise comparativa e o capítulo 7 que demonstra seu uso e benefícios durante a construção de uma aplicação web. Por fim, seguido das considerações finais e referências bibliográficas.

Capítulo 2

Neste capítulo encontra-se uma revisão da literatura com a descrição dos trabalhos e publicações mais representativos referentes ao nosso tema de pesquisa.

Revisão da literatura

Dentre as publicações, temos: artigos de revistas, sites especializadas e documentação científica como artigos, monografias e teses.

2.1 Objetivos das pesquisas

As pesquisas encontradas têm como foco apresentar e comparar as características, a diversidade e os recursos de bibliotecas de componentes JavaServer Faces. Alguns trabalhos se resumem a apresentar, comparar o funcionamento de um leque de componentes de interface [38] [57] [58], outros abordam aspectos que caracterizam a qualidade do framework como um todo.

2.2 Bibliotecas utilizadas e critérios de escolha

Em relação aos critérios de avaliação na escolha dos frameworks, Carmisini, Vahldick [6] utilizaram popularidade, Marchione [31] optou por maturidade e Nóbrega [35] número de usuários. No que diz respeito às bibliotecas adotadas, Costa, Camargo [8], Vaz [57] [58] e Nóbrega [35] utilizaram o Icefaces, Richfaces e Primefaces. Menezes [32] apresenta um estudo entre as bibliotecas de componentes Richfaces, Icefaces e Tamahawke. Carmisini, Vahldick [6]

compararam Apache Tobago, Primefaces e Richfaces. As bibliotecas Myfaces, Icefaces e Richfaces podem ser vistas em Estrada [10]. Enquanto Pitonak [38] realiza uma comparação entre Richfaces, Primefaces, OpenFaces e Icefaces.

2.3 Critérios ou grupos de critérios analisados

Menezes [32] destaca usabilidade, uso do recurso *Asynchronous Javascript and XML* (AJAX), compatibilidade com navegadores, aspecto visual e recurso diferencial como critérios de avaliação. Para Costa, Camargo [8], documentação, infraestrutura e mobilidade são as características que devem ser avaliadas em uma comparação. Número de componentes por funcionalidade, suporte, frequência de atualizações e necessidade de configurações são tratados em Carmisini,Vahldick [6]. Curva de aprendizado é citada em Estrada [10] e desempenho em Marchione [31] e Nóbrega [35].

Para avaliar curva de aprendizado buscaram-se tutorias, manuais, guia do usuário, show cases e necessidade de configurações adicionais. Compatibilidade com navegadores, servidores web, versões e implementações JSF foram analisados para o tópico Infraestrutura. A fim de determinar desempenho, tamanho das respostas geradas foi testado.

2.4 Pontuação e classificação

No intuito de mesurar o nível e a eficiência no atendimento aos critérios estabelecidos regimes de pontuação foram estabelecidos. Em Costa, Camargo [8] utilizou-se uma fórmula própria que envolve a pontuação e o peso do critério analisado bem como um peso para cada grupo de critérios. Somatórios simples das notas são utilizados Menezes [32], Estrada [10] e Nóbrega [35].

As notas em Menezes [32] são distribuídas da seguinte maneira: três pontos para características plenamente atendidas e com bom nível de qualidade; dois pontos características atendidas de forma satisfatória e um ponto quando as características que não são totalmente atendidas ou são atendidas sem um bom nível de qualidade.

A avaliação realizada em Estrada [10] pontua os quesitos qualitativos atribuindo nota zero para resultados menores que 25% ou com resposta igual a muito difícil, pouco. Nota um para resultados maiores ou iguais a 25% e menores

que 50% ou resposta igual a regular, difícil, algo. Nota dois para resultados maiores ou iguais a 50% e menores que 75% ou resposta igual a bom, fácil, bastante. Nota três para resultados maiores ou iguais a 75% ou resposta igual a muito bom, muito fácil, muito.

Nosso trabalho se diferencia dos demais, pois une as vertentes das diversas pesquisas em um só documento. Unimos informações envolvendo critérios de escolha de bibliotecas com documentação, configurações, show case, suporte popularidade, compatibilidade, mobilidade, frequência de atualizações, quantidade de componentes, aspecto visual, produtividade e desempenho.

Esse trabalho não faz a apresentação ou a comparação detalhada de componentes, nosso estudo utiliza alguns componentes apenas para avaliar determinados critérios de qualidade dos frameworks JSF. A comparação realizada nesse documento não tem como finalidade estabelecer o melhor framework mas sim qual se adeque as necessidades do usuário.

Capítulo 3

Este capítulo apresenta a origem do JSF, sua definição, implementações mais conhecidas, arquitetura, termos chaves, ciclo de vida e estrutura básica de uma aplicação.

JavaServer Faces (JSF)

3.1 Origem

O Java está entre as mais populares linguagens de programação [56], é utilizado em um amplo espectro de aplicações, possui três versões:

- *Java Standard Edition* (JAVA SE) – é uma ferramenta de desenvolvimento para a plataforma Java. Ela contém todo o ambiente necessário para a criação e execução de aplicações Java, incluindo a máquina virtual Java (JVM), o compilador Java, as APIs do Java e outras ferramentas utilitárias para uma melhor funcionalidade [9].
- *Java Enterprise Edition* (Java EE) – inclui muitos componentes do Java SE, é adequado para desenvolver aplicações distribuídas em rede em larga escala e aplicativos baseados na web [9].
- *Java Micro Edition* (Java ME) – voltada para o desenvolvimento de aplicações de pequenos dispositivos com limitação de memória como eletrônicos [9].

O JSF é uma especificação que faz parte do Java EE, no lançamento do Java EE6 em dezembro de 2009, passou a ser oficial na plataforma Java EE [5].

JSF é governada por um grupo de peritos Expert Group (EG) que opera sob o Java Community Process (JCP) e representa a comunidade de desenvolvedores JSF e fornecedores de componentes. O EG trabalha dentro do contexto de um Java Specification Request (JSR), a atual especificação JSF é JSR-314, também conhecido como JSF 2. A especificação define a API e um conjunto de componentes de interface do usuário padrão TCK. Ele não inclui detalhes de implementação, essa é uma liberdade que é concedida a qualquer pessoa que queira implementar a especificação [29].

3.2 Definição

É um framework de aplicações web que simplifica o design da interface e separa ainda mais a apresentação da sua lógica de negócio. Um framework fornece bibliotecas e às vezes ferramentas de software que ajudam na construção de sistemas [9-p981].

Embora os componentes JSF padrão sejam suficientes para a maioria das aplicações web mais simples, é possível escrever bibliotecas de componentes personalizadas ou fazer uso de bibliotecas de componentes adicionais fornecidas de forma independente [7-p981].

3.3 Implementações

Atualmente existem duas implementações bem conhecidas: *Mojarra* implementação de referência desenvolvida pela Oracle e *MyFaces* da *Apache Software Foundation*. Estudos mostraram que o número de componentes tem impacto direto sobre um aplicativo baseado em JSF, no caso de estruturas com grandes números de componentes o *MyFaces* apresentou melhor performance com aumento linear sobre a duração do seu ciclo de vida [3].

A primeira versão do JSF (1.0) foi lançada na primeira quinzena de março de 2004, sua especificação é a JSR-127. Depois disso, uma versão de correção de erros (1.1) foi lançada em maio de 2004. Essas duas versões não fazem parte do

conjunto de tecnologias padrões do Java EE. Em Maio de 2006, o JSF 1.2 foi introduzido como JSR-2522, depois disso, três versões avaliação foram liberadas (última em julho de 2008), todos direcionados a Platform J2EE 5.0 [29].

Em julho de 2009 o JSF 2.0 foi lançado seu desenvolvimento seguiu JSR-314 essa versão faz parte da plataforma do Java EE 6. Inúmeras mudanças e melhorias foram adotadas dentre elas: AJAX nativo, navegação implícita e condicional, configuração de *ManagedBeans* com anotação, atribuição de escopos, *converters e validators* com anotação, Facelets como tecnologia padrão de Templates, dois novos escopos adicionados: *View e Custom*, compatibilidade com *Bean Validation* [29].

Em outubro de 2010, foi lançada uma versão de correção 2.1. Em abril de 2013, o JSF 2.2 introduziu novos conceitos como visão de estado, fluxo de página, *viewAction* - validação rápida no servidor, suporte a HTML 5, integração com *portlets*, gerenciar requisições AJAX em fila [29].

3.4 Arquitetura

O JSF adota o padrão arquitetural Model-View-controller (MVC) divide a estrutura da aplicação em três camadas [15]:

- *View* - camada de visualização, responsável pela interface com o usuário e controle das entradas e saídas gráficas e textuais.
- *Model* - controla o comportamento e os dados do domínio da aplicação respondendo as solicitações e instruções para mudar seu estado, além de conter as regras de negócio.
- Controller - controla as solicitações do usuário repassando para a camada *Model* ou para a *View*, adequadamente. Esta divisão em camadas tem o objetivo de aumentar a flexibilidade e a reutilização do código

3.5 Termos chaves

Esta seção contém uma lista de palavras chaves que serão mencionadas nos tópicos seguintes:

- *UI Component* ou Componente de interface é um objeto de estado, mantida no servidor, que fornece funcionalidades específicas para interagir com o usuário final. Componentes de interface do usuário são JavaBeans com

propriedades, métodos, e eventos. Eles são organizados em uma view, que é uma árvore de componentes normalmente exibidos como uma página [30].

- *Renderer* é responsável por exibir um componente UI e traduzir a entrada de um usuário para o valor do componente. *Renderers* podem ser projetados para trabalhar com um ou mais componentes de interface do usuário e um componente de interface pode ser associado com muitos processadores diferentes [30].
- *Validator* ou Validador é responsável por garantir que o valor inserido por um usuário é aceitável. Um ou mais validadores podem ser associados com um componente de interface único [30].
- *Backing beans (managed beans)* são *Java beans* especializadas em recolher valores a partir de componentes de interface do usuário e implementar métodos para evento. Eles também podem conter referências aos componentes de interface do usuário [30].
- *Mensagem* é uma informação que é exibida para o usuário. Apenas sobre qualquer parte da aplicação (*backing beans, validators*, *converters*, e assim por diante) podem gerar informações ou mensagens de erro que podem ser exibidas para o usuário [30].
- Navegação é a capacidade de passar de uma página para outra [30].
- AJAX (*Asynchronous Javascript and XML*) é o uso de tecnologias como Javascript e XML, providas por navegadores, para tornar páginas Web mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações [59].
- JQuery – biblioteca JavaScript que simplifica a escrita de scripts [60].
- Listener ou ouvinte é um objeto que é notificado pela origem do evento (componente ou temporizador) quando um evento ocorre, ele “ouve” um evento e executa um de seus métodos como resposta ao evento [9-p434].
- Faces Context contém todas as informações de estado de um único pedido Java Server Faces e a resposta correspondente. Ele é passado e, potencialmente modificado por cada fase do ciclo de vida do processamento [36].

3.6 Ciclo de vida do JSF

A sequencia de processamento realizada na implementação JSF para a geração das visões está dividido em seis fases [11] conforme ilustrado na figura 1. O fluxo normal de controle é mostrado em linhas sólidas e os fluxos alternativos em linhas pontilhadas:

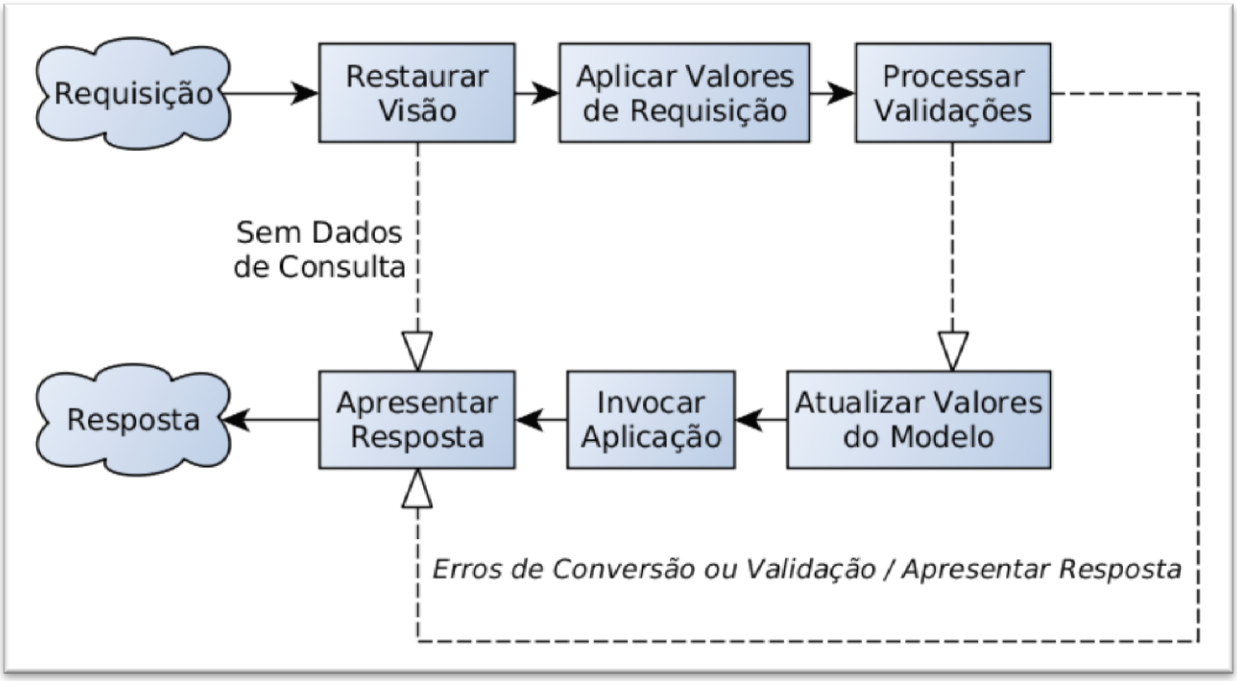

Figura 1 - Ciclo de vida do JSF

Fase 1: Restaurar visão

Caso a página requisitada já tenha sido exibida, é recuperada toda a árvore de componentes para a página requisitada. Caso esteja sendo exibida pela primeira vez é construída uma nova árvore de componentes. No caso da página já ter sido exibida, todos os componentes serão configurados com seu estado anterior. Se a requisição não possuir dados solicitados, a implementação JSF pula para a fase de Renderizar Resposta.

Fase 2: Aplicar Valores de Requisição

Nesta fase o JSF resgata todos os valores informados pelo usuário e os armazena em seus objetos.

Fase 3: Processar Validações

A cadeia de entrada com o valor enviado é convertida para o tipo correto do objeto. Caso ocorra algum erro de validação uma mensagem de erro é adicionada no FacesContext, o componente é marcado como inválido e a implementação JSF invoca a fase de Renderizar Resposta, renderizando a visão ao usuário levando as mensagens de erro. Caso contrário o ciclo de vida continua normalmente.

Fase 4: Atualizar Valores do Modelo

Sua principal atividade é atribuir os valores informados pelo usuário no formulário, para as respectivas propriedades associadas aos *ManagedBeans*. Pode haver erro na conversão, fazendo com que o JSF dispare um erro de tempo de execução, caso ocorra o JSF adiciona esses erros no *Faces-Context* e renderiza a página de visão ao usuário.

Fase 5: Invocar Aplicação

Nesta fase o controlador do JSF chama o método associado ao submeter o formulário, disparando assim a camada de regras de negócio da aplicação. Todos os valores foram validados e carregados nas fases anteriores, por isso poderemos usá-los conforme necessitar. Geralmente é retornada uma string de resultado do método para o JSF efetuar a navegação, se esse valor for *null* o JSF retorna a mesma página que chamou o método.

Fase 6: Apresentar Resposta

Esta fase codifica a resposta e a envia de volta ao navegador.

3.7 Aplicação

Uma aplicação JSF possui a seguinte estrutura [38]:

- Um conjunto de páginas web em que os componentes são definidos;
- Um conjunto de tags para adicionar componentes para a página da web;
- Um conjunto de componente JavaBeans que definem propriedades e funções para componentes em uma página;

- Um descritor de implantação da Web (arquivo web.xml);
- Opcionalmente, um ou mais arquivos de configuração do aplicativo, como um arquivo faces-config.xml, que pode ser usado para definir as regras de navegação de página e configurar os beans e outros objetos, tais como componentes personalizados*;*
- Opcionalmente, um conjunto de objetos personalizados, que podem incluir componentes personalizados, validadores, conversores, ou *listeners*, criado pelo desenvolvedor do aplicativo;
- E um conjunto de *tags* personalizados para representar objetos personalizados na página.

Capítulo 4

Este capítulo apresenta o histórico, dependências e principais características de três bibliotecas JSF: *Richfaces, Icefaces e Primefaces.*

Suítes de componentes JSF

O JSF fornece um conjunto básico de componentes para a criação de aplicações web. No entanto, para atender para atender a demanda dos desenvolvedores por componentes mais sofisticados, existem várias suítes de componentes do JSF ou bibliotecas JSF que seguem o mesmo ciclo e modelo da especificação JSF.
Este capítulo apresenta uma breve introdução de três bibliotecas JSF: *Richfaces, Icefaces e Primefaces.*

4.1 Richfaces

Projeto criado por *Alexander Smirnov*. Em 2005, *Smirnov* foi contratado pela empresa *Exadel* e continuou a desenvolver seu framework chamado de *Ajax4jsf.* A primeira versão do *Ajax4jsf* foi lançada em 2006, ainda no mesmo ano houve a divisão do projeto em *Ajax4jsf* (*opensource* – AJAX em páginas) e *Richfaces* (comercial – AJAX em componentes). Em 2007, *Red Hat e Exadel* assinaram parceria com a *JBoss* que decidiu fundir o *Ajax4jsf* e o *Richfaces* sob o nome *Richfaces*. Isso resolveu problemas de versões e compatibilidades existentes [38].
Richfaces 4 baseia-se no apoio *AJAX, faz uso do JQuery, no controle de eventos [53]* e é padronizado com o JSF2. Essa versão melhorou recursos dinâmicos, de

usabilidade e desempenho possibilitando que todas as melhorias de produtividade do *JSF2* possam ser aproveitadas [53].

Os componentes estão divididos em duas partes [53]:

- A4j – controle de componentes *AJAX* da biblioteca. Servem para submeter dados de forma assíncrona e controlar o comportamento da página nesse tipo de requisição.
- Rich – componentes visuais autônomos da biblioteca com suporte nativo a AJAX. Destinados a exibir, receber e organizar informações na tela.

Em relação a dependências externas, para funcionar o *Richfaces* precisa de quatro arquivos para funcionar: *richfaces-core-api.jar, richfaces-core-impl.jar, richfaces-components-api.jar e richfaces-components-ui.jar* [53].

Entre as características presentes no *Richfaces 4* podemos destacar [35]:

- Controle avançado de requisições *AJAX* baseado em fila para organizar as solicitações assíncronas;
- Validação do lado do cliente, evitando tráfego de rede para o servidor no caso de preenchimento incorreto de campos no formulário;
- Kit próprio para desenvolvimento de componentes (CDK) permite a criação de componentes baseados em *templates* com suporte a AJAX.
- Integração com o *plugin JBoss* Tools para e*clipse* e n*etBeans*.
- Facilidade para criação de testes automatizados em componentes, *action*, *listeners* e páginas.

4.2 ICEfaces

Lançado pela empresa canadense *ICEsoft Technologies em 2004, esta suíte de código aberto e baseada no JQuery e no YahooUI(User Interface Library - YUI). ICEfaces possui três bibliotecas de componentes AJAX [35]:*

- *ICEfaces Advanced Components (ACE) – devem ser preferencialmente utilizados por apresentarem melhor performance. São componentes novos e com baixo tráfego de rede.*
- *ICEfaces ICE Components – ideal para aplicações e navegadores mais antigos, possui os componentes das primeiras versões do ICEfaces que foram atualizados para serem compatíveis com o JSF2 ;*

- *ICEfaces Enterprise Comonents (AEE) – incluída como oferta comercial é composta por mais de 30 componentes com classes mais aprimoradas e recursos e funcionalidades extras. Porém, como dito anteriormente, esses benefícios só são oferecidos* em versões pagas.

Em relação a dependências externas, para funcionar o ICEfaces precisa de quatro arquivos*: Icefaces.jar, icepush.jar, icefaces-ace.jar, icefaces-compat.jar* [25].
A versão mais recente, ICEfaces 3, possui dentre suas características [28]:

- Ajax Automático - eliminando completamente a necessidade de desenvolvedores de, utilizarem o padrão JSF <f:ajax> *tags*;
- Adição de *window scope* – possibilitando o gerenciamento de uma janela ou aba do navegador [35];
- Ajax *push* atualização assíncrona baseada em fila. Permite atualizar de forma incremental qualquer parte da página a qualquer momento, para qualquer grupo de usuários;
- Versão *mobile* para desenvolvimento de aplicações para dispositivos portáteis;
- Integração através de *plugins* com as IDEs e*clipse e netbeans*:

4.3 Primefaces

Surgiu em 2008 através da empresa turca *Prime Tecnhnology*. Seu criador, *Cagatay Civici* é membro do *Java Server Faces Expert Group* e do projeto *Apache MyFaces*. Atualmente oferece dois tipos de versões: Elite e *Community* [35].

A versão Elite permite acesso a um repositório com as atualizações e correções mais recentes, o custo da assinatura anual é de $249 por desenvolvedor.
A versão Pro inclui suporte, onde você pode pedir ajuda diretamente para a equipe do Prime, o custo é feito por orçamento. Outra opção é a case, para criar um novo recurso ou uma correção de um caso particular, sem PRO, estima-se o esforço e o tempo necessário exigido para a solução e faz-se uma oferta com base na complexidade do caso [45].

As versões abertas (*Community*) são releases maiores, e o suporte é oferecido pela comunidade de usuários através de fóruns. Para fazer funcionar o Primefaces, é

necessário incluir em seu projeto apenas o arquivo: primefaces-{versão}.jar sem maiores dependências [45].

A versão *Community* mais recente, *Primefaces* 5, possui dentre suas características [35]:

- Faz uso do *JQuery* e a manipulação de eventos AJAX é baseada na API do JSF2 para requisições assíncronas;
- Assim como o Richfaces, há a possibilidade de uso de skins (temas);
- Permite o emprego do *Primefaces* com outras bibliotecas JSF;
- Paginação por demanda, onde preencher os dados de uma lista de forma fragmentada, apenas após a solicitação do usuário;
- Versão *mobile* para desenvolvimento de aplicações para dispositivos portáteis;
- Suporte a *portlets,* baseado no JSF2 e na API *Portlets2.*

Capítulo 5

Este capítulo apresenta as definições, os métodos e os critérios que serão adotados na análise comparativa. Em seguida temos a execução e os resultados obtidos na análise.

Análise Comparativa

5.1 Definições dos critérios para escolha das bibliotecas

Atualmente existem diversas soluções para a criação de interfaces ricas no mercado: Tobago, Tomahawke, PrettyFaces, JQuery4jsf, OpenFaces, MyFaces, ADFfaces, Primefaces, Icefaces , richfaces entre outras. Para realizar a análise comparativa foram selecionadas bibliotecas que chegaram a um bom nível de maturidade, apresentando bons índices de popularidade e com recursos nivelados [31] [35]. Diante da revisão bibliográfica e dos resultados do site Google Trends [13] referente à popularidade as bibliotecas Richfaces, IceFaces e Primefaces foram selecionadas. O histórico, as dependências e as principais características dessas três bibliotecas JSF foram apresentadas no capítulo anterior.

5.2 Definições dos critérios de análise

Para uma melhor visualização e contextualização a comparação será realizada mediante quatro grupos de critérios, cada grupo apresenta um conjunto de itens que serão analisados.

Grupo	Itens	Descrição do grupo
1.Curva de aprendizado	1. Documentação oficial 2. Documentação não oficial 3. Configurações 4. Show case 5. Suporte 6. Popularidade	Esse grupo prioriza o grau de dificuldade inicial no aprendizado e uso da ferramenta. Irá analisar a disponibilidade de informações existentes na web que reduza sua curva de aprendizado.
2.Infraestrutura	1. Versões do JSF 2. Versões do Java 3. Implementações JSF 4. Suporte ao JSF 2.0 5. Integração com IDE's 6. Compatibilidade com servidores web 7. Compatibilidade com navegadores 8. Compatibilidade com outras bibliotecas de componentes JSF	Esse grupo destaca o suporte oferecido pela biblioteca em relação à infraestrutura. Irá analisar a compatibilidade e suporte em relação ao variado número de versões Java, implementações JSF, servidores, navegadores e outras bibliotecas.
3.Suporte Mobile	1. Suporte a plataforma mobile	Refere-se à ferramenta possuir uma versão mobile
4.Qualidade	1. Frequência de atualizações 2. Problemas encontrados 3. Problemas não resolvidos 4. Variedade de componentes 5. Quantidade de componentes 6. Aspecto visual 7. Produtividade 8. Desempenho	Esse grupo especifica critérios identificadores de qualidade. Verifica se a ferramenta é atualizada e se seus bugs são corrigidos frequentemente. Também analisa a diversidade de componentes, a riqueza de sua interface visual, a simplicidade do código fonte garantindo maior produtividade. O item desempenho, avaliado através de simulação contendo componentes mais complexos, finaliza a lista.

Tabela 1 - Grupo de critérios da análise comparativa

5.3 Descrições dos itens

Grupo 1 – Curva de aprendizado

Itens	Descrição
Documentação oficial	Disponibilidade de tutoriais; guia do usuário; manuais; cursos; vídeo-aulas; idiomas (português e inglês); guia quickstart.

Documentação não oficial	Disponibilidade de livros, tutoriais; cursos e vídeo-aulas.
Configurações	Instalação, necessidade de configurações no arquivo web.xml ; Quantidade e tamanho dos arquivos necessários para funcionamento.
Show case	Organização; usabilidade e documentação integrada.
Suporte	Disponibilidade de fóruns e suporte pago sem necessidade de contrato.
Popularidade	Gráficos do Google Trends referenciando o interesse com o passar do tempo, geral e por região.

Tabela 2 - Descrições dos itens do grupo curva de aprendizado

Grupo 2 – Infraestrutura

Itens	Descrição
Versões do JSF	Versões JSF compatíveis.
Versões do Java	Versões Java compatíveis.
Implementações JSF	Implementações JSF compatíveis: *Mojarra e Myfaces.*
Suporte ao JSF 2.0	Compatibilidades com os novos recursos oferecidos a partir do JSF 2.0.
Integração com IDE's	Integração com as IDE's*: Netbeans e Eclipse*
Compatibilidade com servidores web	Compatibilidade com: *GlassFish (Oracle), Tomcat (Apache) , JBoss (RedHat) e Websphere (IBM).*
Compatibilidade com navegadores	Compatibilidade com *IE, Firefox, Google Chrome, Safari e Opera.*
Compatibilidade com outras bibliotecas de componentes JSF	Compatibilidade entre as bibliotecas escolhidas; Compatibilidade com outras bibliotecas.

Tabela 3 - Descrições dos itens do grupo infraestrutura

Grupo 3 – Suporte mobile

Itens	Descrição
Suporte mobile	Existência de versão mobile estável

Tabela 4 - Descrição do item grupo suporte mobile

Grupo 4 – Qualidade

Itens	Descrição
Frequência de atualizações	Média de atualizações no período de 12 meses.
Problemas encontrados	Obter informações oficiais referentes ao número de problemas encontrados para as versões de cada framework.
Problemas não resolvidos	Obter informações oficiais referentes a casos não resolvidos ou seja em aberto, independente de versão.
Variedade de componentes	Variedade de componentes ofertados por funcionalidade.
Quantidade de componentes	Número total de componentes independente de funcionalidade.
Aspecto visual	Número de temas disponibilizados; Análise do aspecto visual de dois componentes presentes nos três frameworks.
Produtividade	Linhas de código necessárias para criação de componentes; Existência de recursos como: validação, portlets, converters, filtros e paginação.
Desempenho	Análise de tamanho de resposta e taxa de transferência média de respostas do componente dataTable através do software Neoload.

Tabela 5 - Descrições dos itens do grupo qualidade

5.4 O Método de avaliação

A análise comparativa levará em conta os recursos definidos nas tabelas anteriores. Alguns critérios possuem aspectos qualitativos, a fim de estabelecer uma pontuação quantitativa para os critérios serão utilizadas as regras de avaliação da tabela 6 que foi utilizada por Estrada [10].

Quantitativa	0	1	2	3
Qualitativa	Ruim	Regular	Bom	Muito bom
	Muito difícil	Difícil	Fácil	Muito fácil
	Muito pouco	Pouco	Suficiente	Muito
	Não			Sim

Tabela 6 - Regras de avaliação

5.5 O Método de análise

Adotaremos o método de Ponderação Aditiva Simples (Simple Additive Weighting – SAW). Para uma melhor compreensão do método de análise escolhido utilizaremos o exemplo apresentado em Batista [4]. Utilizando como critérios de qualidade: compreensibilidade, precisão, disponibilidade; e como critérios de custo: tempo de resposta e preço. Quanto maiores escores de critérios de qualidade e menores os escores de custo, melhor será a classificação.

Considere cinco fontes de dados hipotéticas (S1,S2,...S5), os valores dos escores, também hipotéticos, de cada critério serão representados por uma matriz de decisão $M = (d_{ij})_{i,j} = 1,...5$. Além da matriz, considere também um vetor de pesos $W = (w1,w2,...w5)$ que reflete a importância individual de cada critério. Os pesos W_j são definidos pelo usuário e podem receber qualquer valor, havendo apenas a obrigatoriedade da soma dos pesos resultarem no valor um. Assim temos:

Matriz de decisão:

$$\begin{array}{c} \\ S1 \\ S2 \\ S3 \\ S4 \\ S5 \end{array} \begin{array}{c} \begin{array}{ccccc} C & P & D & T & Pr \end{array} \\ \begin{pmatrix} 05 & 22 & 20 & 05 & 0,5 \\ 03 & 18 & 99 & 180 & 10 \\ 10 & 10 & 50 & 10 & 00 \\ 03 & 12 & 55 & 03 & 01 \\ 10 & 35 & 10 & 10 & 0,1 \end{pmatrix} \end{array}$$

Onde:

C = critério compreensibilidade;
P = precisão;
D = disponibilidade;
T = tempo de resposta;
Pr = preço

Vetor de Pesos:

$$\begin{array}{c} S1 \\ S2 \\ S3 \\ S4 \\ S5 \end{array} \begin{pmatrix} 18/66 \\ 9/66 \\ 6/66 \\ 22/66 \\ 11/66 \end{pmatrix}$$

Para solucionar um problema com SAW deve-se proceder com três passos: uniformizar os escores para torná-los comparáveis, aplicar os pesos e somar os valores dos escores de cada fonte de dados. Veja as equações aplicadas na uniformização dos escores:

$$V_{ij} = \frac{c_{ij} - c_j^{min}}{c_j^{max} - c_j^{min}}$$

para critérios de qualidade (positivos)

$$Vij = \frac{c_j^{max} - c_{ij}}{c_j^{max} - c_j^{min}}$$ para critérios de custo (negativos)

Onde: c_{ij} é o valor do escore do critério j para a fonte de dados i,
c_j^{max} é o valor máximo do critério j e c_j^{min} é o valor mínimo do critério j.

Para uma determinada fonte de dados Si o seu escore global é calculado pela a soma ponderada definida na equação:

$$V(S_i) = \sum_{J=1}^{5} w_j \mathbf{x} v_{ij}$$ Onde i = 1, 2,...5

Aplicando a equação aos valores de escores todos os valores de escores estarão dentro do intervalo [0,1], sendo os melhores escores de um atributo (critério) com valor próximo de 1 e os piores com valor próximo de 0. Essa propriedade assegura a possibilidade de comparação entre os escores. Assim os cálculos para determinar o ranking das fontes de dados, seriam:

V(S1) = V11 * 18/66 + v12 * 9/66 + v13 * 6/66 + v14 * 22/66 + v15 * 11/66 = 0,7022
V(S2) = V21 * 18/66 + v22 * 9/66 + v23 * 6/66 + v24 * 22/66 + v25 * 11/66 = 0,1818
V(S3) = V31 * 18/66 + v32 * 9/66 + v33 * 6/66 + v34 * 22/66 + v35 * 11/66 = 0,7941
V(S4) = V41 * 18/66 + v42 * 9/66 + v43 * 6/66 + v44 * 22/66 + v45 * 11/66 = 0,5463
V(S4) = V51 * 18/66 + v52 * 9/66 + v53 * 6/66 + v54 * 22/66 + v55 * 11/66 = 0,7751

O método SAW é um método bastante popular e apresenta duas vantagens que é simplicidade e robustez.
Pela análise obtida com o uso do método SAW, o ranking para seleção das fontes hipotéticas ilustradas nesse exemplo seria: 1º S3; 2º S5; 3º S1; 4º S4; 5º S2.

5.5 A comparação

De acordo com o que foi exposto nos itens anteriores desse capítulo, será apresentada agora a análise comparativa dos frameworks RIA. É importante deixar claro que foram utilizadas as versões gratuitas dos frameworks durante a comparação. As informações utilizadas são referentes às versões Richfaces 4.3.7, Primefaces 5 e Icefaces 3.3.0.

De início será apresentada às qualificações alcançadas pelos grupos de critérios e uma argumentação justificando-as. Em seguida cada qualificação será substituída pela sua pontuação correspondente, seguindo assim o método de avaliação adotado.

Grupo 1 – Curva de Aprendizado

Item	Qualificação		
	Richfaces	Primefaces	Icefaces
Documentação oficial	Suficiente	Pouco	Muito
Documentação não oficial	Suficiente	Muito	Suficiente
Configurações	Fácil	Muito fácil	Fácil
Show case	Bom	Muito bom	Muito bom
Suporte	Bom	Muito bom	Bom
Popularidade	Regular	Muito bom	Regular

Tabela 7- Qualificação Grupo Curva de aprendizado

A seguir são descritas as justificativas das qualificações obtidas na análise dos itens da tabela do Grupo 1:

a) Documentação oficial – Nenhum dos frameworks possui versão em português de sua documentação oficial. O Icefaces saiu na frente com uma documentação completa e bastante organizada contendo: manual da API (Interface de programação de aplicativos), guia do usuário, guia *quick start*, cursos do tipo treinamento e vídeo aulas [16][24]. No site do Primefaces e do Richfaces foram encontrados apenas a documentação da API e o manual do usuário, porém o material Richfaces foi mais bem avaliado por ser mais detalhado e explicativo [47][41].

b) Documentação não oficial – em busca de livros foram realizadas pesquisas com o nome dos frameworks no site de compras da Amazon [2] e no resultado consideramos os exemplares que tinham em seu título o nome do framework. O Primefaces trouxe cinco publicações, o Richfaces trouxe três e o Icefaces apenas uma. Em busca de informações diversas, tutoriais e cursos utilizou-se o buscador de conteúdo Google [14] e o site de vídeos Youtube [62], não foram feitas qualquer tipo de restrições na pesquisa. Os resultados podem ser vistos nos gráficos das figuras 2 e 3. De um modo geral o Richfaces e o Icefaces apesar de mais antigos perdem para o primefaces. É importante frisar que boa parte desse material pode estar desatualizado, não utilizando os novos recursos disponibilizados a partir do JSF2. As pesquisas foram realizadas em 16/06/2014.

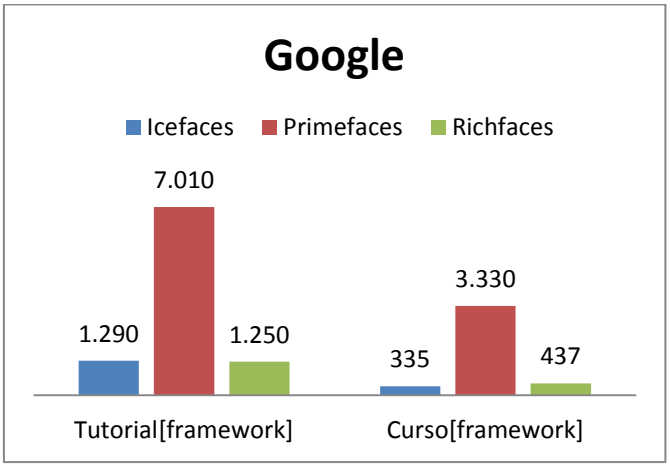

Figura 2 – Gráfico de pesquisa no Google

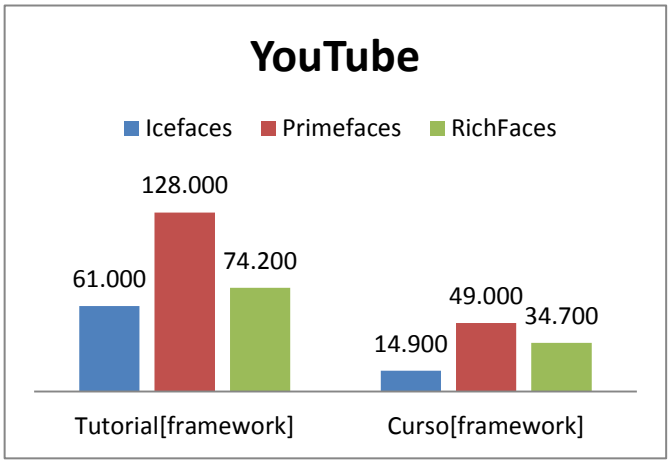

Figura 3 – Gráfico de pesquisa no You tube

c) Configurações – Nenhum dos frameworks exige algum tipo de instalação, é necessário apenas fazer o download dos arquivos indicados no site e/ou manual do usuário. Segundo o manual, o Richfaces [47] necessita de sete arquivos para funcionar e o Iceface necessita de quatro [25]. Ambos precisam da inserção de algumas linhas de configuração no arquivo web.xml, enquanto que o Primefaces requer apenas um arquivo e não necessita de qualquer tipo de configuração [40]. Em relação ao tamanho dos arquivos, a soma em megabytes dos arquivos do Icefaces 7,64, Richfaces é de 9,13 já do Primefaces se resume a 2,5.

d) Show case – Todas as opções apresentam ótima organização e excelente usabilidade. É fácil encontrar o modelo e o código fonte de um determinado componente. Um diferencial nesse caso, apresentado pelos frameworks Primefaces e Icefaces, foi a integração do manual do usuário ao show case [18][49][42] .

e) Suporte – Destacam-se nesse item os fóruns que possuem um grande número de colaboradores e postagens. Em 20/06/2014, de acordo com as informações contidas nos fóruns podemos afirmar que o Icefaces possui um total de 171.180 membros e 67.114 postagens, enquanto o Primefaces apresenta 37.581 membros e 114.350 postagens. Não foram localizadas essas informações no fórum do Richfaces [21] [44] [50]. No quesito suporte pago, mesmo para versões gratuitas, o Primefaces oferece em seu site novas funcionalidades, através de orçamentos estimados por esforço e complexidade da solução [45]. O site do Icefaces disponibiliza suporte profissional com serviços de consultoria e desenvolvimentos de componentes personalizáveis, porém apenas para versão comercial [23].

f) Popularidade – Para analisar esse item, em 21/06/2014 foram utilizados gráficos do Google Trends [13]. Veja na fig 4 o interesse por cada framework com o passar do tempo, e nas figuras 5, 6 e 7, o número de pesquisas de

forma global e por região. Observa-se pelos gráficos que o interesse no Richfaces e Icefaces é desproporcional ao crescimento em busca de informações sobre o Primefaces.

Figura 4 - Gráfico Interesse com o passar do tempo

Figura 5 - Gráfico interesse regional Richfaces

Figura 6 - Gráfico Interesse regional Icefaces

Figura 7 - Gráfico interesse regional Primefaces

Grupo 2 – Infraestrutura

Item	Qualificação		
	Richfaces	Primefaces	Icefaces
Versões do JSF	Muito bom	Muito bom	Muito bom
Versões do Java	Bom	Muito bom	Bom

Implementações JSF	Muito bom	Muito bom	Muito bom
Suporte ao JSF 2.0	Sim	Sim	Sim
Integração com IDE's	Sim	Sim	Sim
Compatibilidade com servidores web	Muito bom	Muito bom	Muito bom
Compatibilidade com navegadores	Bom	Muito bom	Bom
Compatibilidade com outras bibliotecas de componentes JSF	Sim	Sim	Sim

Tabela 8 - Qualificação Grupo Infraestrutura

A seguir são descritas as justificativas das qualificações obtidas na análise dos itens da tabela do Grupo 2, tendo como fonte de informação os releases notes [52] [17], guia do desenvolvedor [47] [25] [40] e fóruns [21] [44] [50] de cada framework :

a) Versões do JSF – Todos os frameworks são compatíveis com as versões 2.0, 2.1 e 2.2 do JSF. Caso um projeto tenha uma versão antiga do JSF é possível fazer uma migração seguindo os tutoriais disponíveis na documentação das bibliotecas.

b) Versões do Java – O Primefaces é compatível com versões a partir do Java 5, enquanto que o Icefaces e o Richfaces afirmam que suas versões são compatíveis a partir do Java 6.

c) Implementações JSF – Verificamos que os frameworks são compatíveis com as duas principais distribuições JSF Mojarra e Myfaces.

d) Suporte ao JSF2 – Todas as bibliotecas possuem suporte aos novos recursos que surgiram a partir da versão 2.0 do JSF.

e) Integração com IDE's – É possível utilizar o Eclipse e o Netbeans como ambiente integrado para desenvolvimento de aplicações através dos plugins disponibilizados pelo Icefaces e do JBoss Tools para o Richfaces. Para o Primefaces é necessário apenas adicionar a biblioteca dentro do projeto.

f) Compatibilidade com servidores web – é possível utilizar em qualquer dos frameworks um dos servidores compatíveis com Java Platform, Enterprise Edition 6 (JEE6), como o JBoss Application Server 7, como Apache Tomcat 7, Glassfish 3.1.2.2 ou Websphere 8.5.5.

g) Compatibilidade com navegadores – Nesse item verificamos se o framework possui compatibilidade, independente do número de versões compatíveis, com os seguintes browsers: Internet Explorer, Mozzila Firefox, Google Chrome, Safari e Opera. Primefaces é compatível com todos. O Icefaces e o Richfaces não são compatíveis com o Opera.

h) Compatibilidade com outras bibliotecas de componentes JSF – as bibliotecas são compatíveis entre si e com outras bibliotecas de componentes JSF. Para utilizar mais de uma biblioteca em um mesmo projeto é necessário fazer alterações no arquivo web.xml e não se recomenda utilizar componentes de bibliotecas diferentes em uma mesma página.

Grupo 3 – Suporte Mobile

Item	Qualificação		
	Richfaces	Primefaces	Icefaces
Suporte a plataforma mobile	Sim	Sim	Sim

Tabela 9 - Qualificação Grupo Suporte Mobile

A seguir são descritas as justificativas das qualificações obtidas na análise dos itens da tabela do Grupo 3, tendo como fonte de informação o site dos frameworks [19] [46] [48]:

a) Suporte a plataforma mobile – esse item refere-se à existência de uma versão mobile gratuita disponibilizada pelo fabricante. Todos frameworks atendem esse requisito.

Grupo 4 – Qualidade

Item	Qualificação		
	Richfaces	Primefaces	Icefaces
Frequência de atualizações	Muito Bom	Bom	Bom
Problemas resolvidos	Muito Bom	Bom	Regular
Problemas em aberto	Ruim	Bom	Bom
Variedade de componentes	Bom	Muito bom	Bom
Quantidade de componentes	Regular	Muito bom	Regular
Aspecto visual	Bom	Muito Bom	Muito Bom
Produtividade	Bom	Bom	Bom
Desempenho	Bom	Bom	Regular

Tabela 10 - Qualificação Grupo Qualidade

A seguir são descritas as justificativas das qualificações obtidas na análise dos itens da tabela do Grupo 4, tendo como fonte de informação ferramentas de gerenciamento de projeto [22] [51] [39] e os *showcases* [18] [49] [42] de cada framework:

a) Frequência de atualizações – No que tange gerenciamento de projetos, a ferramenta Jira é utilizada pelo Richfaces e o Icefaces enquanto que o Primefaces utiliza o Google Code. Baseado nas datas de lançamento das versões é possível afirmar que o Icefaces e o Primefaces possuem uma média de dois lançamentos por ano. Enquanto que o Richfaces faz atualizações com maior frequência, em 2013 cinco versões foram lançadas. Essa informação é de grande importância, pois indica o tempo médio para correções de bugs e inclusão de novos componentes e funcionalidades nas versões, nesse quesito o Richfaces se saiu melhor que seus concorrentes.

b) Problemas resolvidos – Nesse item verificou-se o número de problemas resolvidos para cada framework e suas respectivas versões que estão sendo analisadas. Quanto menor o valor melhor a qualificação. Para tornar mais justa a comparação tendo em vista que o Richfaces realiza atualizações frequentes, somamos os problemas das versões 4.3.4 (lançada em 23/09/2013), 4.3.5 (lançada em 27/01/2014), 4.3.6, (lançada em 11/04/2014)

e 4.3.7 (lançada em 27/05/2014). O objetivo dessa soma é encontrar um valor aproximado para o mesmo intervalo de tempo. Assim, é possível observar na fig. 8 que o número total de casos ou *issues* do framework Icefaces é bem maior que o do Richfaces.

Figura 8 - Problemas resolvidos Icefaces e Richfaces

Tendo em vista que a ferramenta de gerenciamento do Primefaces difere em aspectos com a dos outros frameworks seus dados são apresentada na fig 9. A versão do Primefaces tratou 289 casos, desses 151 eram defeitos, 51 aprimoramentos e 87 novas funcionalidades.

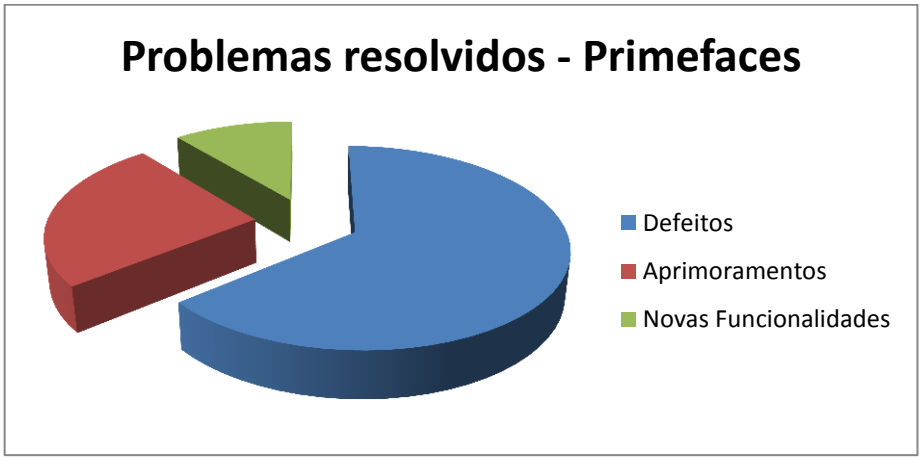

Figura 9 - Problemas resolvidos Primefaces

c) Problemas em aberto – Nesse item buscamos os casos em aberto para cada framework independente de versão. A grande maioria desses casos está relacionada com versões futuras, ou seja, ainda não foram lançadas. Na fig. 10 é possível ver que o número de casos em aberto do Richfaces é muito maior se comparado aos outros dois frameworks. Boa parte dos casos em aberto no Richfaces refere-se à versão 3.x.

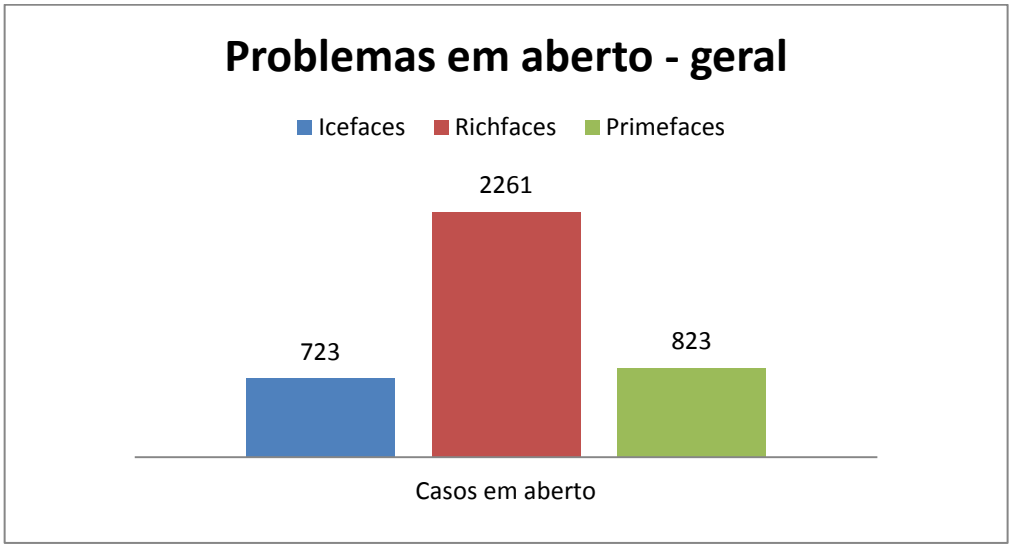

Figura 10 - Casos em aberto

d) Variedade de componentes – Através dos showcases de cada framework constatamos que todos apresentam os componentes básicos e podem atender sem problemas o desenvolvimento de aplicações simples. É possível encontrar ainda componentes de multimídia para exibição de vídeos e áudio, galeria de imagens, efeitos do tipo animação e arrastar/soltar. Estão presentes apenas no Primefaces e no Icefaces os recursos de mapas e criação de gráficos. O componente que permite a captura de imagem via webcam foi encontrado apenas no Primefaces. Sem dúvida um diferencial do Primefaces é oferecer a seus usuários a possibilidade de escolher para determinada funcionalidade diversos componentes diferentes.

e) Quantidade de componentes – Nesse item utilizamos as informações contidas no *showcase* de cada framework. O Richfaces apresenta sessenta e um, o Icefaces setenta e o Primefaces cento e quarenta. É bastante significativa a

superioridade do Primefaces nesse quesito quando comparado com seus concorrentes.

f) Aspecto visual – Em 2012, houve a integração de diversos componentes do Primefaces por parte do framework Icefaces [26]. No blog oficial do Primefaces há um post afirmando que o Icefaces copiou o Primefaces linha por linha [43]. Polêmicas a parte, é perceptível a semelhança entre alguns componentes dos dois frameworks.

Em relação a temas o Icefaces oferece vinte e seis opções enquanto o Richfaces possui apenas sete opções. O primefaces disponibiliza trinta e oito diferentes tipos de temas pré-definidos. Para fechar esse item escolhemos dois componentes para compará-los visualmente. Menubar e data table, presentes nos três frameworks e frequentemente utilizados na maioria das aplicações.

As figuras 11 e 12 apresentam as barras de menu do Icefaces e Primefaces respectivamente. São esteticamente elegantes e agradáveis.

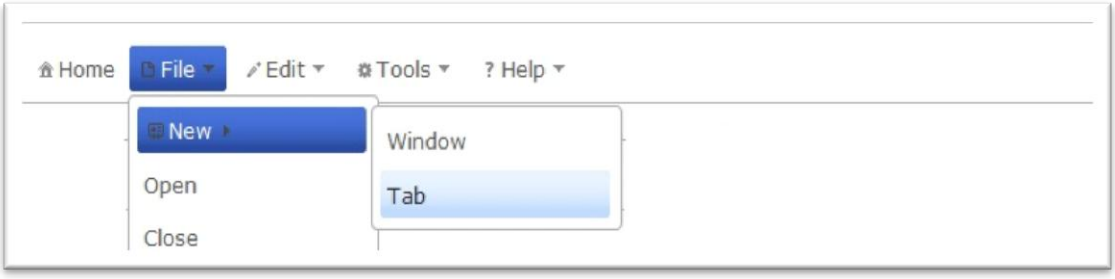

Figura 11 - Barra de Menu Icefaces

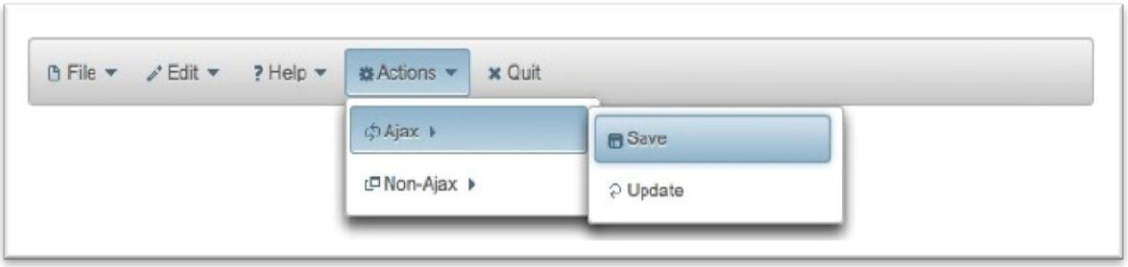

Figura 12 - Barra de Menu Primefaces

A barra de menu do Richfaces pode ser vista na fig. 13 e seu data table na fig. 16. Mais simples possui um visual que esteve presente por vários anos em aplicações e sistemas operacionais, mas que hoje está continuamente sendo substituído e modernizado.

Figura 13 - Barra de Menu Richfaces

Novamente é notória a semelhança entre o data tables do Icefaces que pode ser visto na fig 14, e do Primefaces apresentado na fig 15. O diferencial do primefaces nesse caso ficou pelas diferentes opções de filtro.

ID	Nome	Chassis	Peso	Accel	MPG	Custo
1	Enduro	Van	£ 15.383.	10	17.86	$ 6,617.17
2	Tomcat	Ônibus	£ 7.331.	15	16,65	$ 31,464.24
3	Doublecharge	Levantamento	£ 5.333.	15	17,84	$ 10,922.73
4	Peixe-espada	Ônibus	£ 10.956.	5	5.17	$ 6,019.83
5	Iguana	Levantamento	£ 1.696.	10	9.43	$ 19,736.16
6	Dardo	Motocicleta	£ 9.261.	15	12.85	$ 37,947.84
7	Peixes	Luxo	£ 7.846.	10	15.13	$ 19,235.20
8	Flash	Mid-Size	£ 11.499.	10	12.74	$ 29,942.38
9	Tomcat	Mid-Size	£ 10.766.	15	7.04	$ 14,342.74
10	Peixes	Subcompacto	£ 2.082.	10	13.38	$ 8,015.01

1 2 3

Figura 14 - Data Table Icefaces

Search all fields: Enter keyword

Id	Year	Brand	Color	Status	Price
		Select One	Colors	All Sold Sale	
60b42250	1987	Jaguar	Red	Sale	$66,193.00
37109b0a	1993	Volkswagen	Maroon	Sold	$96,533.00
af078f08	2004	Volkswagen	Brown	Sold	$96,970.00
cd3607ea	1986	Mercedes	Maroon	Sale	$76,252.00
a56e6f49	1995	Mercedes	Maroon	Sold	$76,242.00
ce10afbf	1960	Mercedes	White	Sold	$92,124.00
03d349f1	1977	Volvo	Brown	Sale	$11,171.00
42ff1cb9	1983	Volkswagen	Black	Sale	$97,973.00
b891070c	1985	Ford	Red	Sale	$30,165.00
d42d8296	1994	Audi	Green	Sale	$91,373.00
contains	lte	exact	in	equals	custom (min)

Figura 15 - Data Table Primefaces

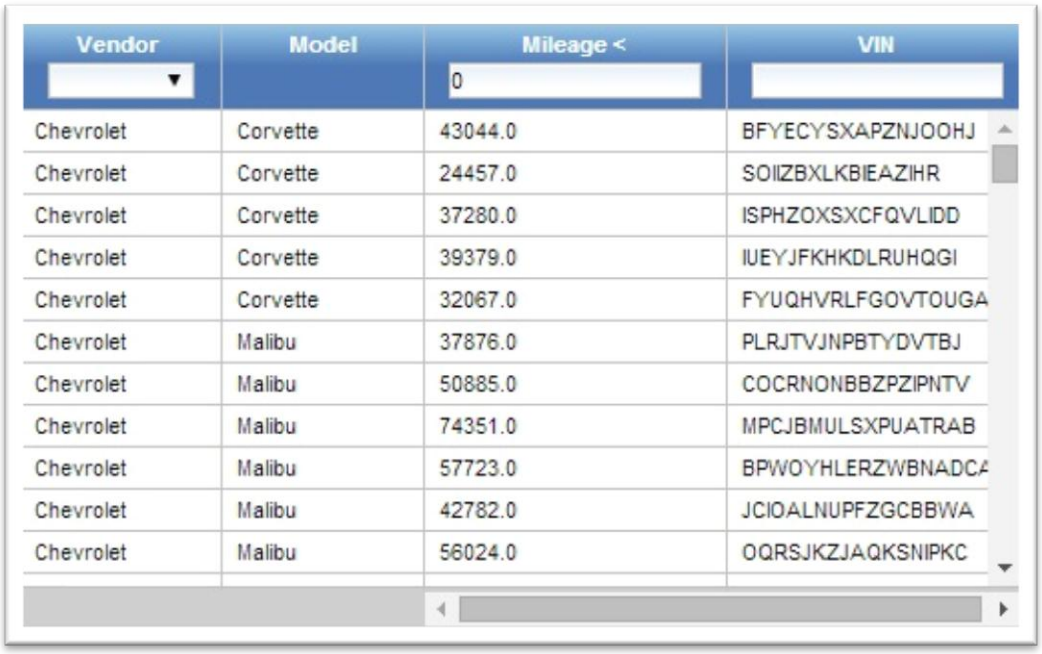

Figura 16 - Data Table Richfaces

g) Produtividade – No que diz respeito ao código fonte que cria os componentes, é possível perceber através dos exemplos fornecidos nos show cases dos frameworks que não há muitas diferenças. Seja em relação à quantidade de linhas ou da forma que elas são estruturadas dentro da página web suas aplicações são bem semelhantes. Todos os frameworks oferecem também recursos que aumentam a produtividade facilitando a criação de filtros, *convertes*, paginação, validação e listeners. Apenas o recurso portlet não é fornecido na versão gratuita do Icefaces [19].

h) Desempenho – esse item comparou os resumos estatísticos de indicadores globais de desempenho do componente dataTable, modelo mais simples, exibindo cem registros trazidos da base de dados. Para isso, utilizamos a ferramenta de teste Neoload [34].

   Os detalhes da realização desses testes e dos materiais e métodos utilizados estão no Apêndice A deste documento. A comparação contendo os resultados obtidos nos testes de desempenho pode ser vista na tabela11.

Resultados obtidos com o Neoload			
	Richfaces	Icefaces	Primefaces
Rendimento total	11,59MB	58,44MB	14.22MB
Vazão média	0,77Mb/s	3,86Mb/s	0,94Mb/s

Tabela 11 - Resultados obtidos com Neoload

Onde:

- Rendimento total é a soma dos tamanhos das respostas a todos os pedidos.
- Vazão Média é a taxa de transferência média de respostas do servidor (a jusante).

Note que em relação ao tamanho da resposta e sua taxa de transferência média o rendimento do Icefaces é bem pior que dos seus concorrentes. É importante ressaltar que a análise desse componente tem o objetivo de realizar uma análise inicial dessas bibliotecas. Para determinar a biblioteca com o melhor desempenho um estudo centralizado e mais detalhado testando diversos componentes precisaria ser realizado.

Pontuação alcançada:

			Pontuação		
Grupo	Itens	Peso	Richfaces	Primefaces	Icefaces
1	Documentação oficial	7/127	2	1	3
	Documentação não oficial	7/127	2	3	2
	Configurações	5/127	2	3	2
	Show case	7/127	2	3	3
	Suporte	5/127	2	3	2
	Popularidade	5/127	1	3	1
2	Versões do JSF	3/127	3	3	3
	Versões do Java	3/127	2	3	2
	Implementações JSF	3/127	3	3	3
	Suporte ao JSF 2.0	7/127	3	3	3
	Integração com IDE's	7/127	3	3	3
	Compatibilidade com servidores web	4/127	3	3	3
	Compatibilidade com navegadores	7/127	2	3	2

	Compatibilidade com outras bibliotecas de componentes JSF	3/127	3	3	3
3	Suporte a plataforma mobile	2/127	3	3	3
4	Frequência de atualizações	5/127	3	2	2
	Problemas resolvidos	5/127	3	2	1
	Problemas em aberto	7/127	0	2	2
	Variedade de componentes	7/127	2	3	2
	Quantidade de componentes	7/127	1	3	1
	Aspecto visual	7/127	2	3	3
	Produtividade	7/127	2	2	2
	Desempenho	7/127	2	2	1

Tabela 12 - Pontuação alcançada

Aplicando o método SAW

- Matriz de decisão

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23
Richfaces	02	02	02	02	02	01	03	02	03	03	03	03	02	03	03	03	03	00	02	01	02	02	02
Primefaces	01	03	03	03	03	03	03	03	03	03	03	03	03	03	03	02	02	02	03	03	03	02	01
Icefaces	03	02	02	03	02	01	03	02	03	03	03	03	02	03	03	02	01	02	02	01	03	02	02

- Uniformização dos critérios

$$V_{ij} = \frac{C_{ij} - C_j^{min}}{C_j^{max} - C_j^{min}}$$

Onde: C_{ij} é o valor do escore do critério j para a fonte de dados i, C_j^{max} é o valor máximo do critério j e C_j^{min} é o valor mínimo do critério j. Considere C_j^{min} como zero e C_j^{max} como três.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23
Richfaces	0,7	0,7	0,7	0,7	0,7	0,3	1	0,7	1	1	1	1	0,7	1	1	1	1	0	0,7	0,3	0,7	0,7	0,7
Primefaces	0,3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0,7	0,7	0,7	1	1	1	0,7	0,7
Icefaces	1	0,7	0,7	1	0,7	0,3	1	0,7	1	1	1	1	0,7	1	1	0,7	0,3	0,7	0,7	0,3	1	0,7	0,3

Vetor de pesos

S1	7/127
S2	7/127
S3	5/127
S4	7/127
S5	5/127
S6	5/127
S7	3/127
S8	3/127
S9	3/127
S10	7/127
S11	7/127
S12	4/127
S13	7/127
S14	3/127
S15	2/127
S16	5/127
S17	5/127
S18	7/127
S19	7/127
S20	7/127
S21	7/127
S22	7/127
S23	7/127

- Escore Global

$$V(S_i) = \sum_{j=1}^{23} W_j \mathbf{x} V_{ij} \qquad i = 1,2 \text{ e } 3$$

$V(S_1)$ = 0,7*0,05 + 0,7*0,05 + 0,7*0,04 + 0,7*0,05 + 0,7*0,04 + 0,3*0,04 + 1*0,02 + 0,7*0,02 + 1*0,02 + 1*0,06 + 1*0,06 + 1*0,03 + 0,7*0,05 + 1*0,02 + 1*0,01 + 1*0,04 + 1*0,04 + 0*0,06 + 0,7*0,06 + 0,3*0,06 + 0,7*0,06 + 0,7*0,06 + 0,7*0,06 = 0,69

$V(S_2)$ = 0,3*0,05 + 1*0,05 + 1*0,04 +1*0,06 + 1*0,04 + 1*0,04 + 1*0,02 + 1*0,02 + 1*0,02 + 1*0,06 + 1*0,06 + 1*0,03 + 1*0,05 + 1*0,02 + 1*0,01+ 0,7*0,04 + 0,7*0,04 + 0,7*0,06 + 1*0,06 + 1*0,06 + 1*0,06 + 0,7*0,06 + 0,7*0,06 = 0,88

$V(S_3)$ = 1*0,05 + 0,7*0,05 + 0,7*0,04 + 1*0,05 + 0,7*0,04 +0,3*0,04 + 1*0,02 + 0,7*0,02 + 1*0,02 + 1*0,06 + 1*0,06 + 1*0,03 + 0,7*0,05 + 1*0,02 + 1*0,01 + 0,7*0,04 + 0,3*0,04 + 0,7*0,06 + 0,7*0,06 + 0,3*0,06 + 1,0*0,06 + 0,7*0,06 + 0,3*0,06 = 0,72

Resultado do Ranking utilizando o método SAW:

- Primefaces
- Icefaces
- Richfaces

Gostaríamos de deixar claro que o objetivo desse trabalho não é definir qual a melhor biblioteca afinal todas elas já possuem maturidade e são boas opções para aumentar a produtividade no desenvolvimento web. Nosso objetivo é orientar profissionais para que escolham o framework que melhor se adeque as suas necessidades, adotar critérios e atribuir pesos de acordo com sua importância possibilitou-nos alcançar esse objetivo. O próximo capítulo apresenta uma ferramenta que automatiza esse processo.

Capítulo 6

Este capítulo apresenta a ferramenta de comparação criada tendo como base o estudo realizado no capítulo cinco.

Ferramenta de comparação

6.1 Apresentação

O objetivo dessa ferramenta é realizar a comparação entre bibliotecas de JSF, nela será possível escolher as bibliotecas que serão comparadas e estabelecer os pesos de acordo com as necessidades e interesses do usuário. Não é necessário desse usuário um amplo conhecimento das técnicas JSF ou de aplicações web tendo em vista que os critérios definem de forma global e de alto nível os recursos concretos e profundos necessários para uma aplicação independente de sua complexidade. Assim o perfil de usuário para a ferramenta engloba de um estudante a um especialista em desenvolvimento de software.

6.2 Ambiente de desenvolvimento

Foram utilizados os seguintes materiais para a criação da ferramenta:

- Eclipse Kepler versão 4.3 como IDE de desenvolvimento,
- Mysql (mysql-connector-java-5.1.5-bin.jar) para gerenciar dados,

- Java 7 como linguagem de programação,
- Tomcat 7.0 como servidor de aplicações web,
- Java Server Faces 2.1 (Mojarra 2.1.6) como framework de aplicação web,
- Primefaces 5 como framework de componentes.

6.3 Requisitos

Aqui é realizada uma análise detalhada dos requisitos funcionais exigidos na aplicação. Fluxo de eventos:

1. O usuário acessa a página do sistema destinado à análise comparativa e clica no botão “Gerar Comparação”.
2. Em uma segunda página o usuário seleciona os frameworks que devem ser comparados através de um componente listbox.
3. Ainda na mesma página o usuário poderá alterar os pesos da lista de critérios contida no componente datatable.
4. O usuário clica no botão Enviar.
5. Uma mensagem é exibida contendo o nome dos frameworks selecionados e suas respectivas notas.

6.4 Modelagem

Utilizaremos no desenvolvimento da ferramenta o paradigma de programação orientada a objetos e o padrão MVC. O diagrama de classe é o diagrama central desse tipo de modelagem. Nele podemos visualizar a estrutura e as relações das classes que servirão de modelo para objetos.

A fig 17 representa o diagrama de classe, perceba que existem quatro pacotes com cores diferentes. O pacote amarelo representa a interface e contém as páginas index e comparação, esse pacote se comunica exclusivamente com o pacote *control* em azul onde está localizada a classe CriterioBean. Essa classe associa-se com o pacote *model* em verde bem como o pacote *dao* em marrom.

No pacote ***model*** há duas classes que descrevem os objetos da aplicação enquanto no pacote *dao* armazena as classes que fazem acesso ao banco de dados.

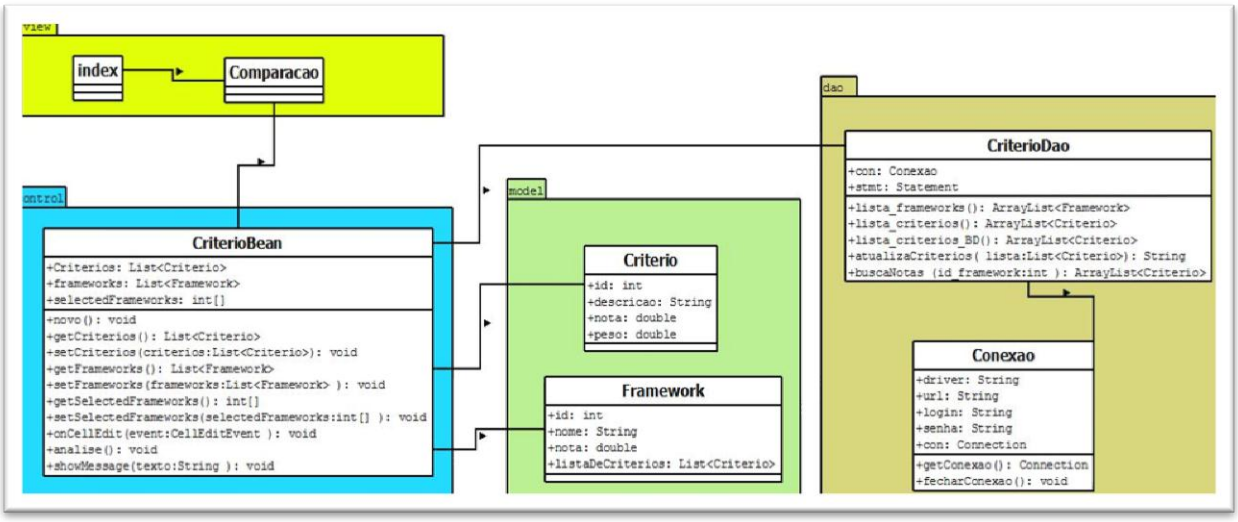

Figura 17- Diagrama de classe (Ferramenta)

Definido o diagrama de classe, se criou o modelo lógico da base de dados representado pela fig.18. Esse modelo mostra os componentes de cada tabela, as ligações entre as tabelas e suas chaves primárias e estrangeiras. Perceba através da imagem que a tabela nota possui uma chave estrangeira composta da chave primária da tabela critério e da chave primária da tabela framework. A tabela critério_bd contém os pesos que são sugeridos e a tabela critérios armazena os pesos com as alterações realizadas pelo usuário.

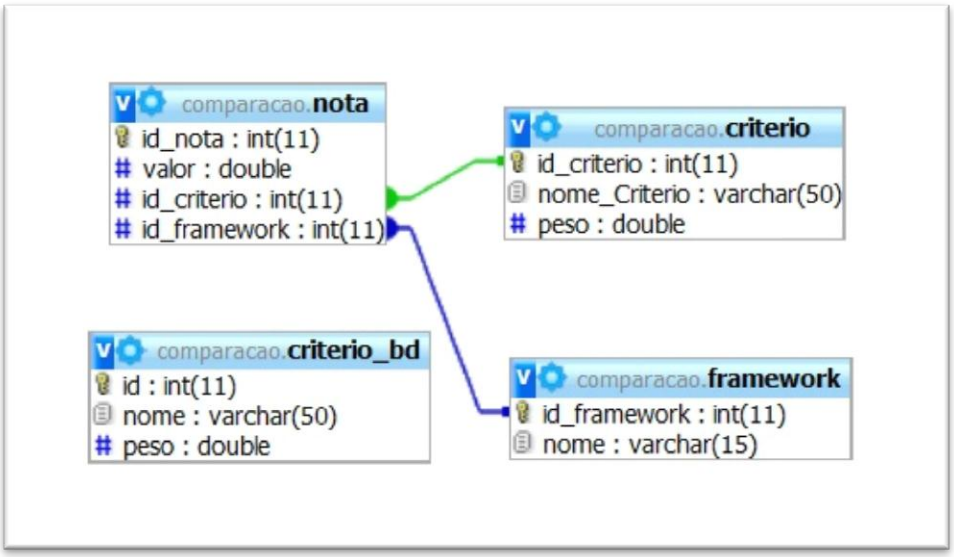

Figura 18 - Modelo lógico (Ferramenta)

6.5 Desenvolvimento

Definido o requisito, o diagrama de classes, a modelagem do banco de dados e concluídos a instalação e configuração do ambiente de desenvolvimento. Iniciou-se o desenvolvimento do projeto, com a criação dos pacotes, classes e páginas.

Dentro da pasta src criou-se os pacotes e suas respectivas classes de acordo com o diagrama de classe construído. A estrutura final do projeto pode ser vista na fig.19. O conteúdo completo de cada classe e página xhtml podem ser vistos no anexo B deste documento. As imagens das telas são apresentadas nas figuras 20, 21 e 22.

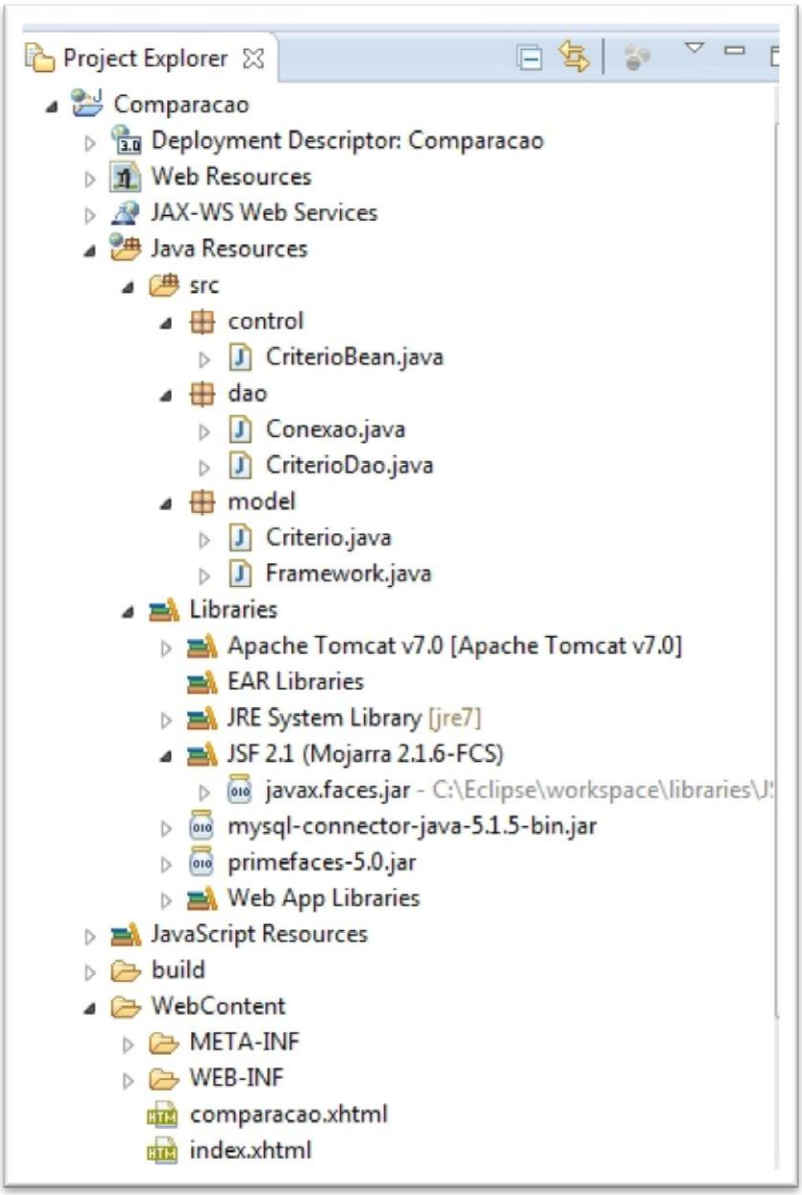

Figura 19 - Estrutura do projeto

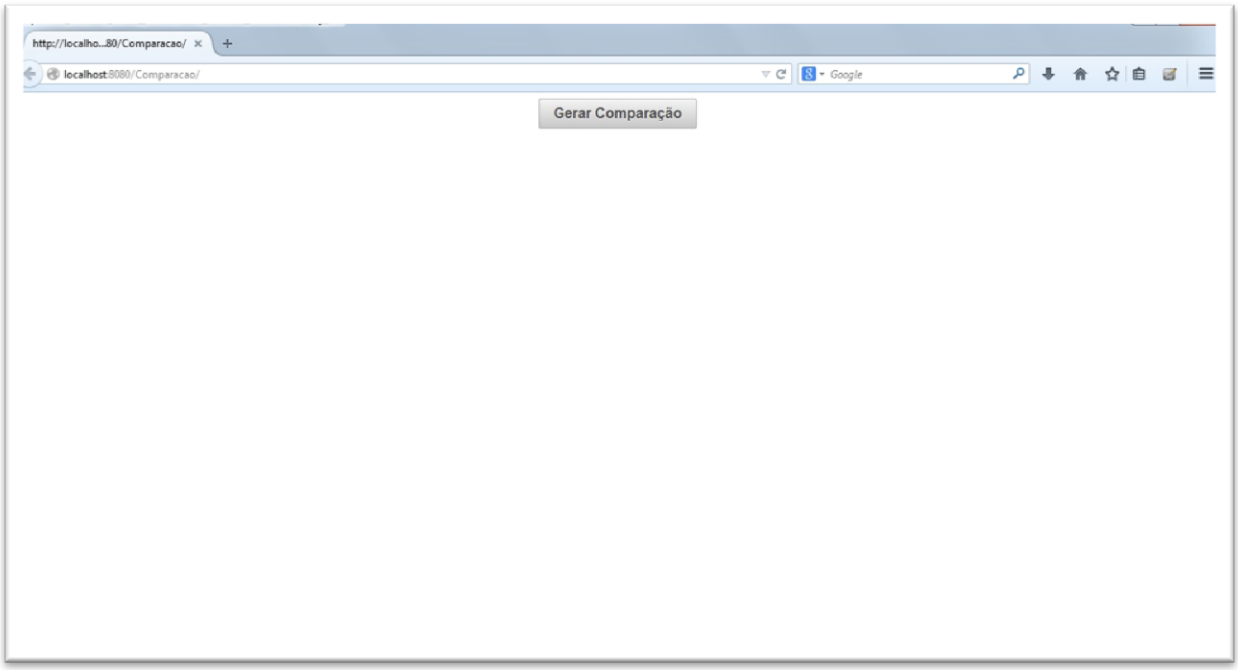

Figura 20 - Tela da Comparação: Início

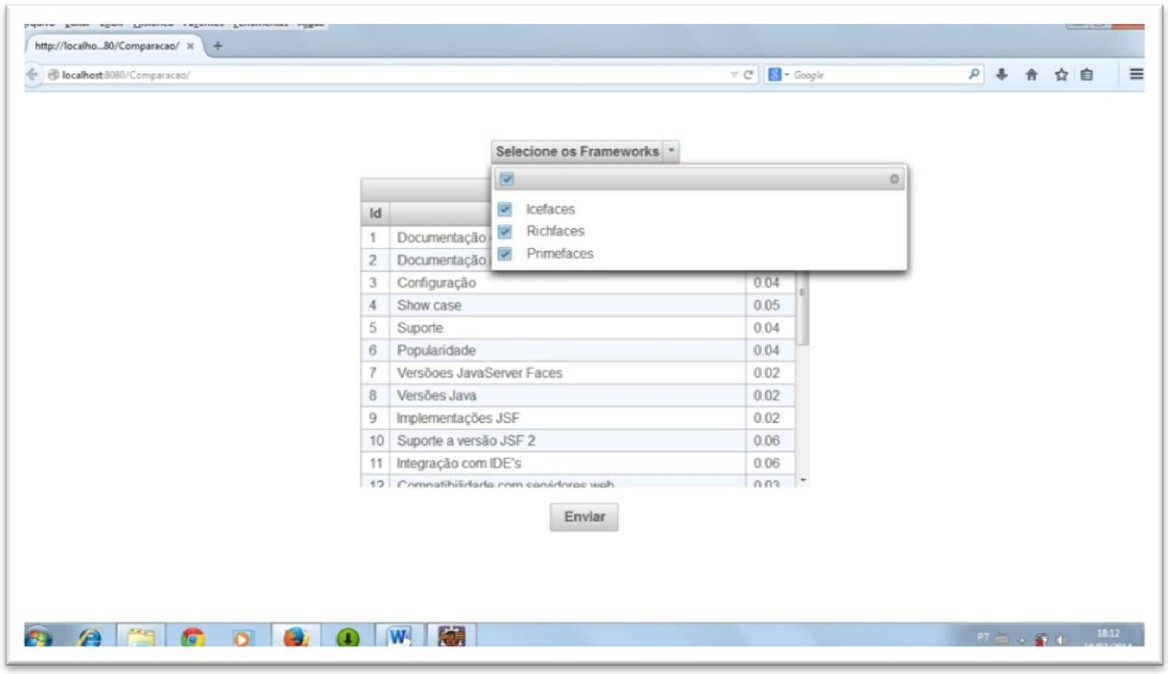

Figura 21 - Tela da Comparação: Seleção dos frameworks

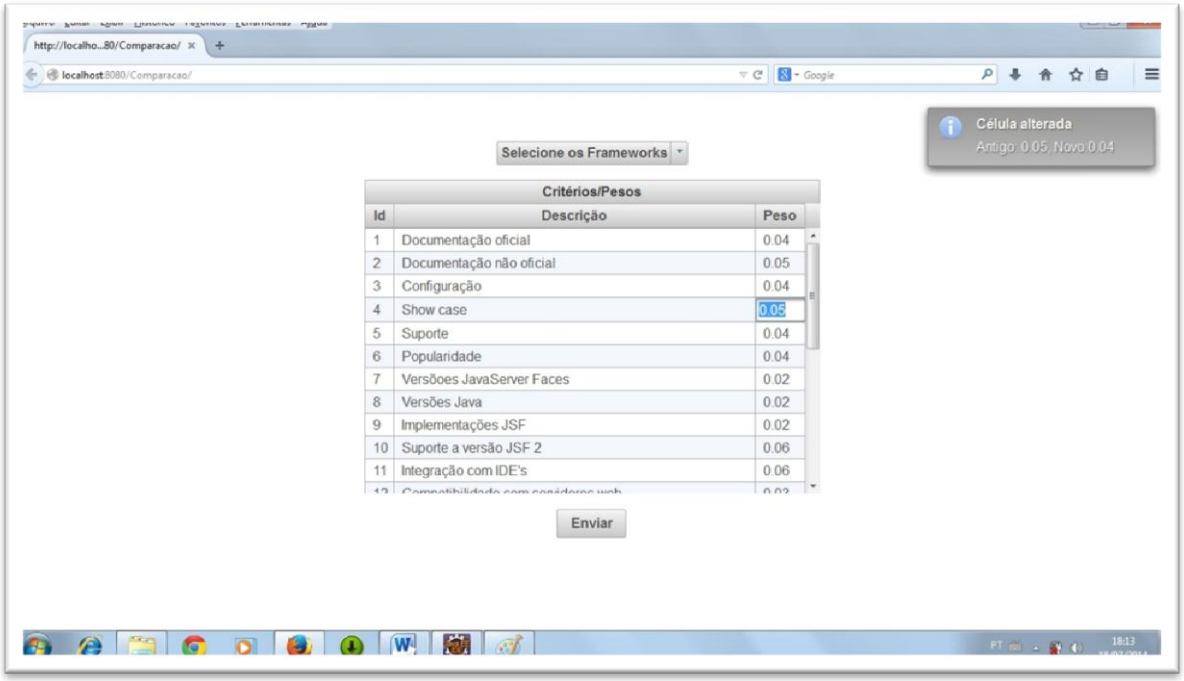

Figura 22 - Tela da comparação: Alteração dos pesos

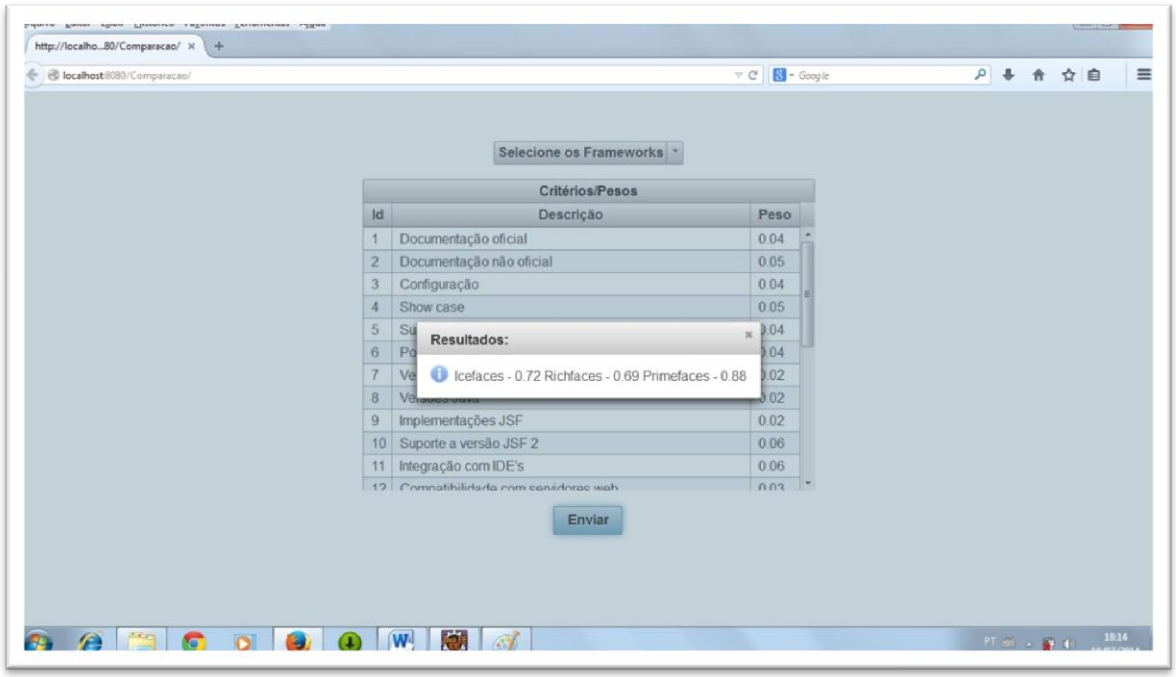

Figura 23 - Tela Comparação: Mensagem com o resultado

Capítulo 7

Este capítulo tem irá validar a ferramenta criada no capítulo seis através de seu uso em um problema real.

Aplicação em um caso real

7.1 Caracterizações das empresas

Duas empresas serão os cenários no estudo de caso do presente trabalho, usaremos nomes fictícios para preservar as empresas tendo em vista que não nos foi autorizada a divulgação de seus dados originais.

A primeira empresa será chamada aqui de Destaque localizada na cidade do Recife, é de pequeno porte possuindo menos de dez funcionários. Trabalha com a gestão de documentos, oferecendo a armazenagem, preservação e rastreabilidade de documentos.

A segunda empresa nomeada por nós como VipTechnology, também de pequeno porte, tem em sua equipe atualmente quinze profissionais. Atua no mercado de Tecnologia da Informação oferecendo soluções em softwares.

7.2 O problema

A empresa Destaque possui um sistema próprio que realiza o controle das caixas e de seus documentos armazenados. A empresa deseja fornecer a seus clientes uma ferramenta que lhes permita acesso às caixas armazenadas. O sistema precisa ser web e possuir uma interface amigável com filtros de pesquisas e

listagens dinâmicas. Além de recursos de segurança como controle de acesso e de informação.

A empresa VipTecnology, contratada para desenvolver uma solução para o problema, está com sua equipe de desenvolvedores sobrecarregada com outras atividades e por isso um profissional com pouca experiência em interfaces web precisou assumir o projeto. Para facilitar no desenvolvimento e conseguir cumprir o prazo de entrega de três semanas, o profissional foi orientado a utilizar frameworks e suítes de componentes para ajudá-lo no desenvolvimento.

7.3 Ambiente de desenvolvimento

Foram utilizados pela empresa VipTecnology para o desenvolvimento da aplicação:

- Ferramenta de análise comparativa de frameworks JSF;
- Eclipse Kepler versão 4.3 como IDE de desenvolvimento,
- Mysql (mysql-connector-java-5.1.5-bin.jar) para gerenciar dados,
- Java 7 como linguagem de programação,
- Tomcat 7.0 como servidor de aplicações web,
- Java Server Faces 2.1 (Mojarra 2.1.6) como framework de aplicação web,

Os pesos carregados inicialmente pela ferramenta de análise comparativa e apresentados no capítulo cinco deste documento são exatamente os valores estabelecidos pelo desenvolvedor da Empresa VipTecnology.

O maior peso utilizado foi 0,06. Ele foi estabelecido para quase todos os itens do grupo qualidade com exceção de frequência de atualizações e problemas resolvidos que receberam 0,05. Por ser inexperiente na criação de interfaces para web o profissional também estipulou em 0,05 os pesos do grupo curva de aprendizado. A redução no valor dos pesos se deu nos itens: versões Java, implementações JSF, compatibilidade com outras bibliotecas e versão mobile. Isso se deve pelo fato do suporte a esses recursos não serem de grande importância para o desenvolvedor no momento. Diante dos pesos estabelecidos a ferramenta indicou como melhor opção o framework Primefaces 5.

7.4 Requisitos

Listaremos a seguir os requisitos funcionais mínimos exigidos para a versão beta do sistema:

1. Layout utilizando logomarca da Empresa.
2. Login identificando o usuário e a empresa.
3. Listagem contendo todas as caixas pertencentes à empresa do usuário logado.
4. Os itens id, armazém e conteúdo devem ser exibidos em colunas o restante dos dados em uma componente do tipo caixa de dialogo.
5. São essenciais os recursos de paginação e filtros na tabela para facilitar a pesquisa de dados para o usuário.
6. Incluir opções de cadastrar, alterar e excluir caixas da empresa do usuário logado cujo depósito seja diferente da Empresa Destque.

7.5 Desenvolvimento

Seguindo os padrões de projetos utilizados pela empresa adotou-se o paradigma de programação orientada a objetos e o padrão MVC. A estrutura da pasta src contendo as classes criadas e a pasta WebContent com as páginas web podem ser visualizadas respectivamente nas fig.23 e 24.

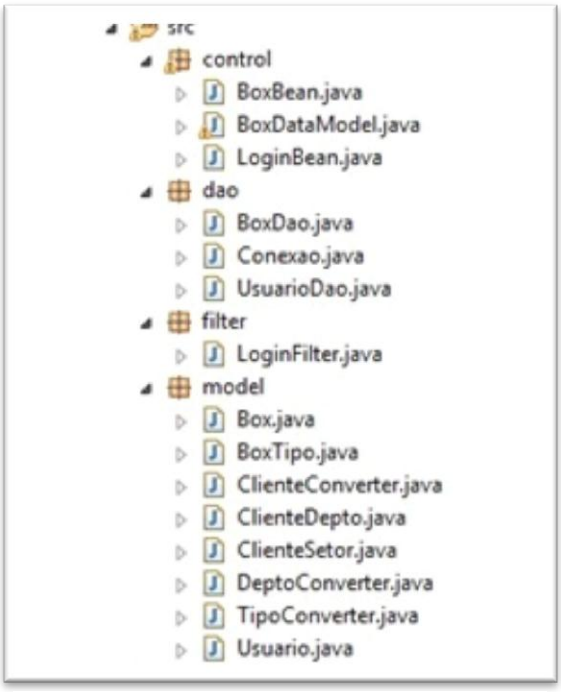

Figura 24 - Classes do Projeto

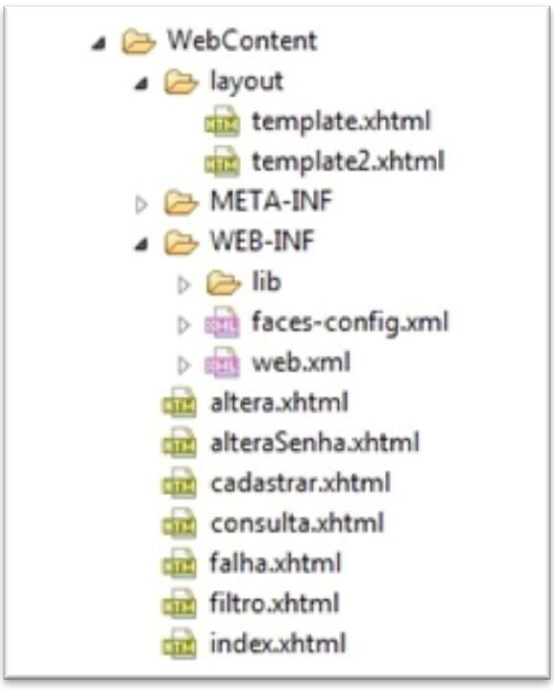

Figura 25 - Páginas web do Projeto

As imagens seguintes trazem as telas criadas com o Primefaces. Na fig.25 temos a tela de login do usuário onde é possível observar validação e mensagem de retorno.

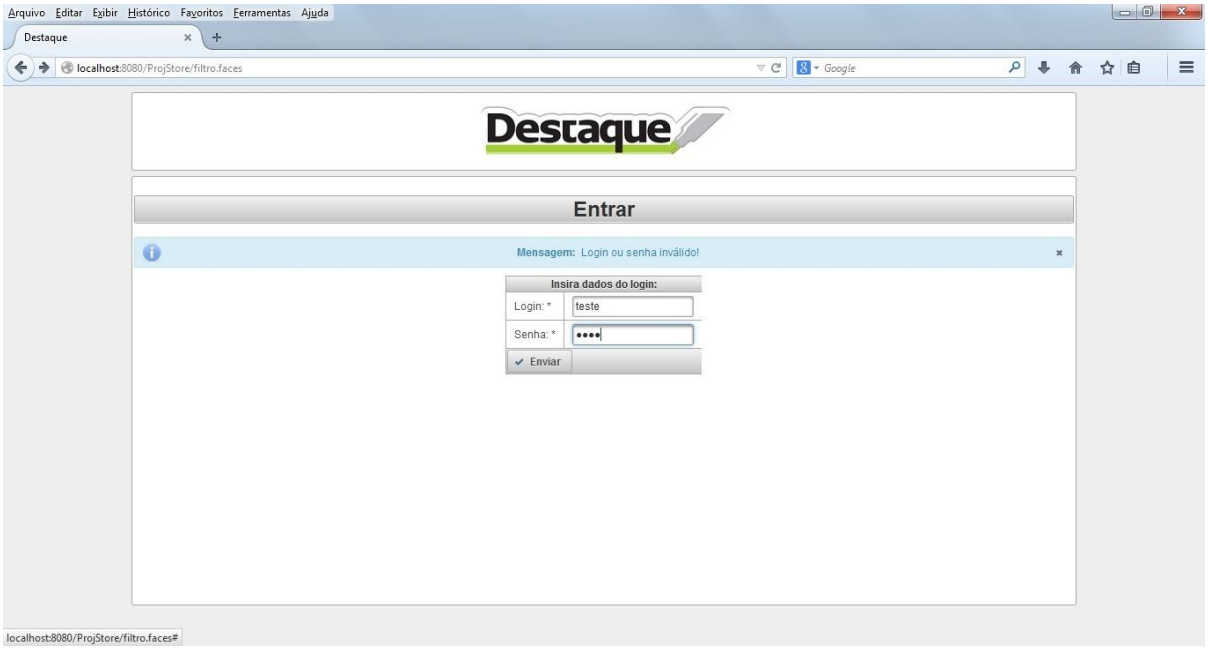

Figura 26 - tela de login

A fig.26 apresenta a primeira tela logo após o usuário efetuar seu login. Quando uma linha dessa tabela é selecionada automaticamente a mensagem com as demais informações do objeto é exibida. A paginação indica em azul a página atual.

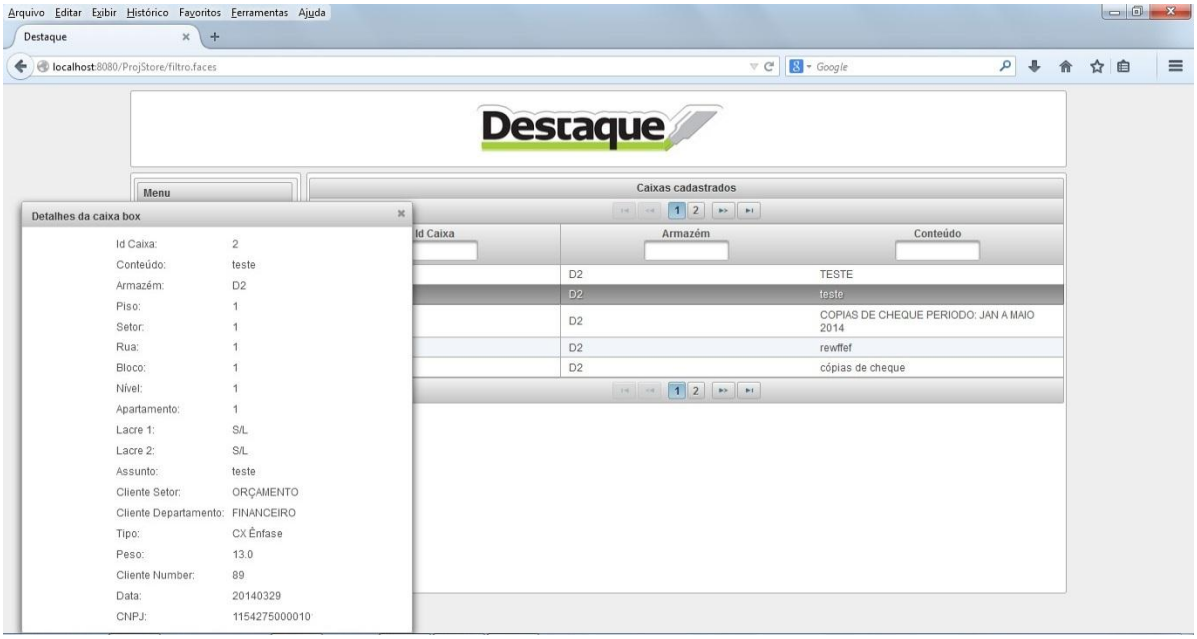

Figura 27 – Tela com Seleção, Paginação e Diálogo

O filtro exigido pelo cliente foi inserido e seu funcionamento está registrado na fig.27.

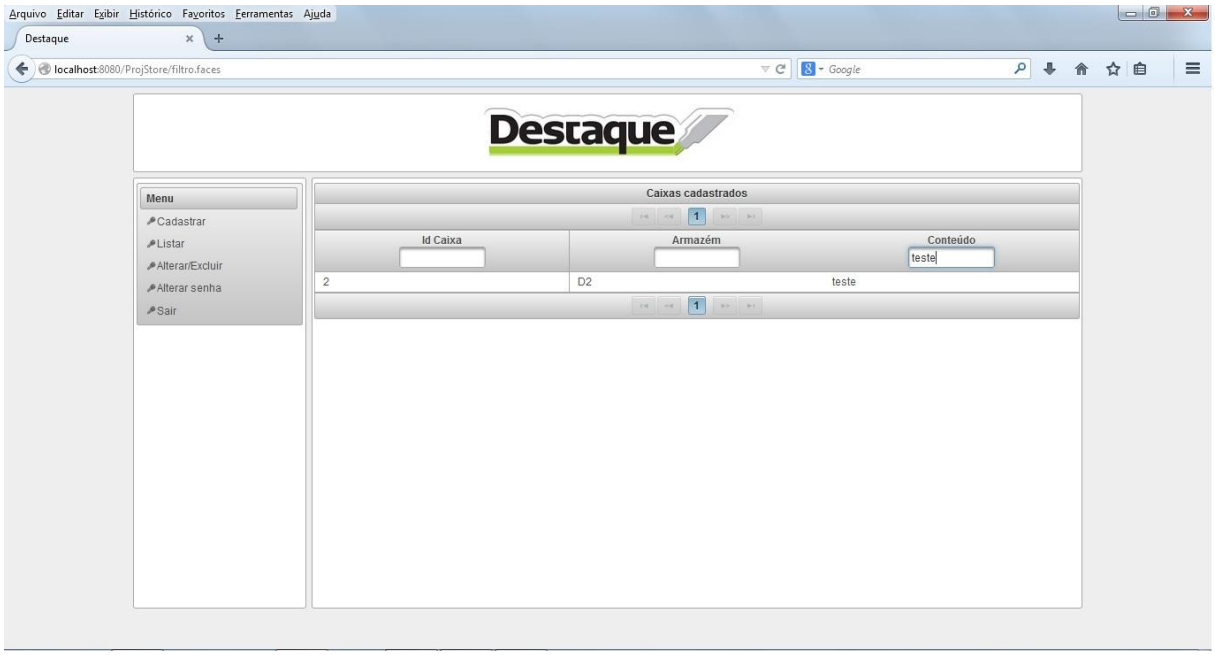

Figura 28 – Tela com Filtro

A fig.28 checa o depósito e o CGC do cliente da caixa a ser alterada ou excluída, caso não seja permitida a operação uma mensagem de erro é exibida.

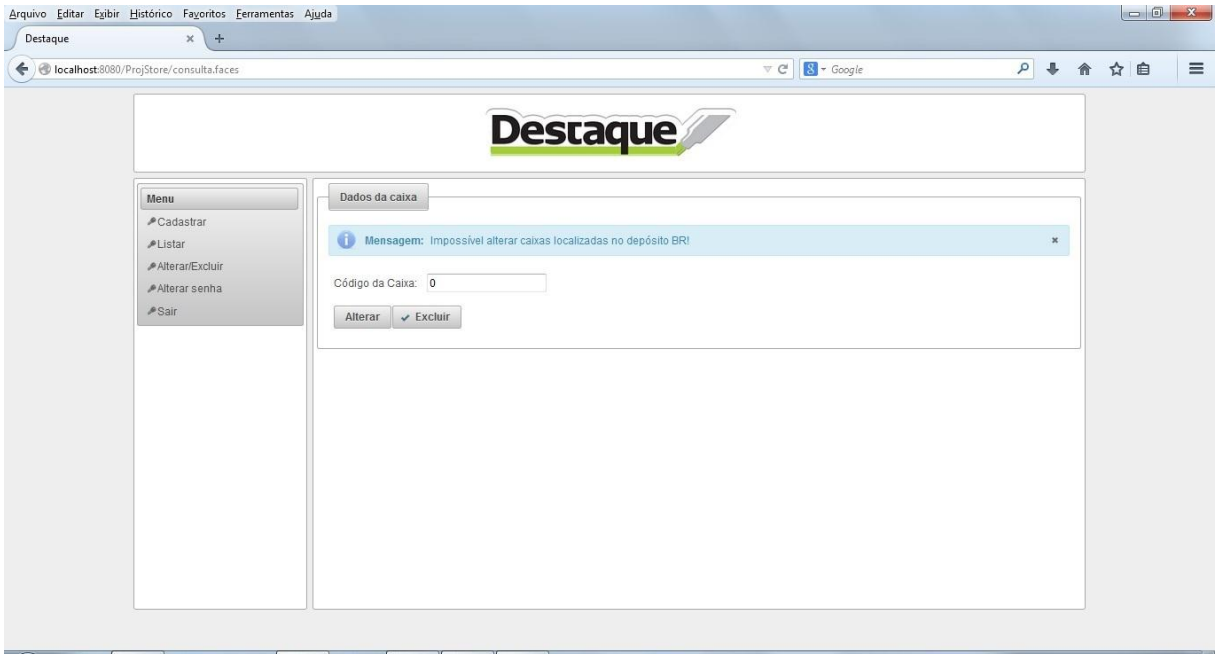

Figura 29 - Tela para Alterar e Excluir

Um formulário que pode ser visto na fig.28 formado por diversos componentes compõe a tela de cadastro e de alteração.

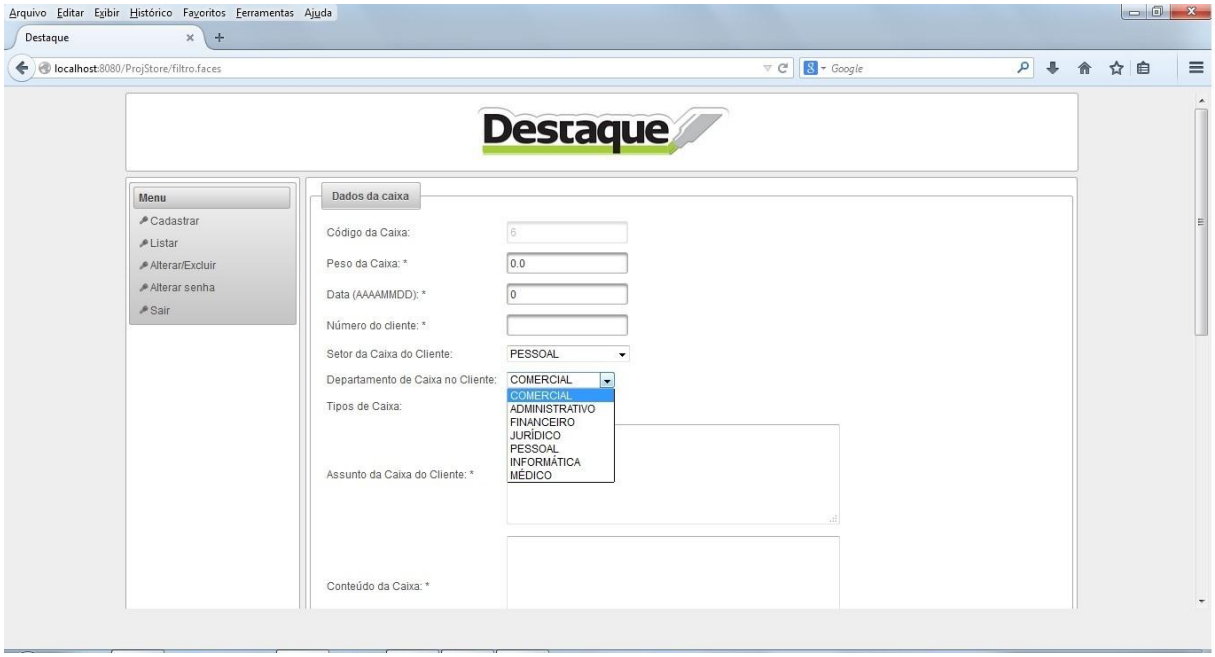

Figura 30 - Tela de cadastro

Apresentamos as telas criadas, porém não entraremos em detalhes em relação ao código fonte deste projeto tendo em vista que nosso objetivo nesse momento é a ferramenta de comparação e os resultados obtidos com sua indicação. Assim, vamos relatar agora os pontos ressaltados pelo profissional em relação à ferramenta.

7.6 Feedback

Para analisar os resultados alcançados com a ferramenta, realizamos um questionário contendo algumas perguntas para o desenvolvedor que a utilizou.

De acordo com Merriam [63], a pesquisa qualitativa é um conceito “guarda-chuva” que cobre diversas formas de investigação que nos ajudam a entender e explicar o sentido dos fenômenos sociais com o mínimo de ruptura dos contextos naturais.

1) Qual a sua expectativa em relação ao projeto a ser desenvolvido? Você acha que a ferramenta teve influência em relação aos resultados alcançados?

A inexperiência na criação de interfaces para web e a necessidade de criar uma aplicação que atendesse os requisitos através de componentes visualmente elegantes e eficientes fez crer que o prazo de três semanas não seria cumprido. Todavia isso não aconteceu, acredita-se que a ferramenta de comparação tem seu crédito nesse resultado.

2) Você teve alguma dificuldade em utilizar a ferramenta? Acha que os critérios apresentados por ela conseguiram atender de forma eficiente suas necessidades?

Não houve qualquer dificuldade na utilização da ferramenta. Os critérios apresentados não exige nenhum conhecimento prévio do JSF ou das bibliotecas de componentes. São de fácil compreensão e trazem um leque que cobre os recursos JSF, as necessidades do desenvolvedor e de uma aplicação de qualidade.

3) A ferramenta teve influência positiva em relação aos resultados alcançados? Ressalte onde e como ocorreu.

A ferramenta poupou tempo de pesquisa visto que uma seleção prévia das bibliotecas do mercado é realizada já direcionando o usuário para os mais populares e utilizados frameworks: Richfaces, Primefaces e Icefaces. Sem ela seria necessário investigar as possibilidades existentes no mercado e não teria como saber se a escolha seria a mais acertada.

Permitiu identificar e relacionar os critérios de acordo com a importância deles. A maior preocupação era o tempo disponível para aprender o framework e criar a aplicação do cliente garantindo a qualidade do produto final.

Evitou que uma escolha inadequada colocasse em risco a produtividade e possibilitou o direcionamento para um framework que conseguiu atender as necessidades desse projeto.

4) Se a ferramenta teve influência de forma negativa em relação aos resultados alcançados, ressalte onde e como ocorreu?

Não houve nenhum aspecto negativo, não percebeu-se qualquer tipo de situação onde a ferramenta tenha prejudicado ou atrapalhado o andamento do projeto. Enfim, ela deve ser vista como um recurso agregador que deve colaborar e guiar os profissionais da área de desenvolvimento de software para a adoção de frameworks de interfaces JSF.

5) Você voltaria a utilizar a ferramenta ou a indicaria para seus colegas de trabalho?
Sim, sem dúvidas.

Considerações Finais

O primeiro objetivo desse trabalho foi realizar uma análise comparativa entre frameworks JSF, para isso criamos uma variedade de critérios grupo de critérios que foram agrupados em quatro grupos: Curva de aprendizado, infraestrutura, mobile e qualidade. Cada framework foi analisado sob a ótica desses critérios recebendo uma qualificação que se transformaria em nota. De posse das notas dos componentes e utilizando o método SAW de ponderação igualamos os scores, estabelecemos um vetor de pesos e realizamos o somatório que forneceu a nota final obtida por cada framework.

As notas finais obtidas vieram a corroborar com tudo que foi exposto neste trabalho, mostrando que o RichFaces , Primefaces e o ICEFaces são boas escolhas mas apresentam resultados diferentes sob diversos aspectos. No presente trabalho, desenvolveu-se um estudo entre três frameworks de JSF, escolhidos com base em popularidade, este estudo trouxe subsídios para realizar uma análise apresentando aspectos variados como documentação, variedade de componentes, suporte, compatibilidades com servidores e navegadores, frequência de atualizações entre outros.

Um dos problemas encontrados no estudo foi o acesso a informações, os releases notas do Primefaces que possui o Google Code como ferramenta de configuração dificulta um pouco o acesso às informações o Jira utilizado pelo Icefaces e Richfaces disponibiliza as informações de forma mais clara. Foi através do Jira que se verificou o alto número de casos em aberto por parte do Richfaces.

A documentação do Primefaces está centralizada no guia do usuário, o Richfaces e Icefaces possuem uma documentação mais abrangente sobre os componentes e suas funcionalidades.

Em relação à diversidade e número de componentes o Primefaces ganha destaque, o Icefaces em 2010 inseriu diversos componentes do Primefaces em sua biblioteca, o que melhorou bastante sua galeria embora muitos vejam isso de forma negativa.

No quesito desempenho, utilizando a ferramenta Neoload e um dataTable simples, a versão do framework Icefaces apresentou o tamanho de resposta bem maior que seus concorrentes.

De um modo geral, o PrimeFaces superou o Richfaces e o Icefaces em alguns critérios ou houve um equilíbrio entre os três frameworks.

O segundo objetivo foi criar uma aplicação que guiasse o desenvolvedor de forma técnica para a adoção do framework JSF ferramenta que melhor se adeque às suas necessidades. Nossa meta foi produzir uma aplicação web que executasse a comparação, nela o usuário estabelece os pesos para cada critério de acordo com sua conveniência para o projeto a ser desenvolvido.

O terceiro e último objetivo foi validar a ferramenta utilizando-a como parte da solução para um problema real. Nessa fase uma empresa foi contratada para desenvolver uma aplicação web e o profissional que ficou responsável em liderar o projeto não tinha conhecimentos sobre como construir interfaces web. A ferramenta de comparação foi utilizada com o objetivo de orientar o profissional para a melhor escolha de acordo as necessidades particulares daquela situação.

Tão importante quanto direcionar as atividades de profissionais de desenvolvimento de software é minimizar o esforço e o tempo gasto nelas. A ferramenta de comparação tem como proposta ser um recurso que vai agilizar o processo de escolha entre um grupo de ferramentas obtendo o melhor resultado possível. De fácil compreensão ela auxilia na avaliação de determinados pontos que precisam de prioridade em relação ao demais, isso facilita o trabalho da equipe e reduz o tempo e por consequência o custo gasto nos projetos.

Por ser um estudo inicial, faz-se necessário definir aspectos futuros que ampliem e tragam informações sob a ótica de diversos profissionais aperfeiçoando e dando embasamento a pesquisa. As qualificações e as notas dadas nesse estudo são de uma visão única e isso deve acrescer e globalizar entre diversos profissionais de diversos perfis e níveis.

Em relação à adoção de biblioteca de componentes ela se mostrou algo muito benéfico, permitindo que uma parte do esforço gasto com a construção das interfaces, fosse abstraída, dando aos programadores mais tempo para se dedicarem às lógicas de negócios envolvidas nos projetos.

As bibliotecas são indicadas tanto para organizações quanto para estudantes e profissionais com pouca experiência em criar interfaces para web. O uso corporativo pode ser beneficiado com a possibilidade de cursos e suporte oficial por parte de seus criadores.

O estudo destacou a utilidade prática e a importância de se fazer uma avaliação com base em uma lista de critérios permitindo assim que a organização consiga listar todas as suas necessidades e pontuá-las. A análise baseada nas perspectivas estratégicas e técnicas são essenciais para escolher a melhor solução com mais segurança.

APÊNDICES

Apêndice A. Teste de desempenho das bibliotecas Richfaces, Icefaces e Primefaces.

Para realizar os testes utilizamos os seguintes materiais:

- Eclipse Kepler versão 4.3 como IDE de desenvolvimento,
- Mysql (mysql-connector-java-5.1.5-bin.jar) para gerenciamento de dados
- Java 7 como linguagem de programação
- Tomcat 7.0 como servidor de aplicação web,
- Java Server Faces 2.1 (Mojarra 2.1.6) como framework de desenvolvimento
- Icefaces 3.3.0 como framework de componente
- Primefaces 5 como framework de componente
- Richfaces 4.3.7 como framework de componente
- Plugin do Icefaces para o Eclipse (IF-3.3.0-IM-1.3.0-Eclipse-4.3-plugins-B) disponível no site do icefaces [20].
- Plugin do Richfaces para o Eclipse – Jboss Tools Kepler 4.1.2 instalado através do Eclipse Marketplace.
- Neoload 4.2 como ferramenta de análise de desempenho

Realizamos os downloads necessários, efetuamos a configuração do Icefaces [27] e do Richfaces [47] no Eclipse. Em seguida criamos um projeto para cada biblioteca [27][12] [55].

Em cada projeto criamos dentro da pasta src três pacotes: control, dao e model.

A classe Contato foi criada no pacote model. Em control, criamos ContatoBean. Dentro do pacote Dao inserimos Conexao e ContatoDao. A estrutura final da pasta src pode ser vista na fig. 30.

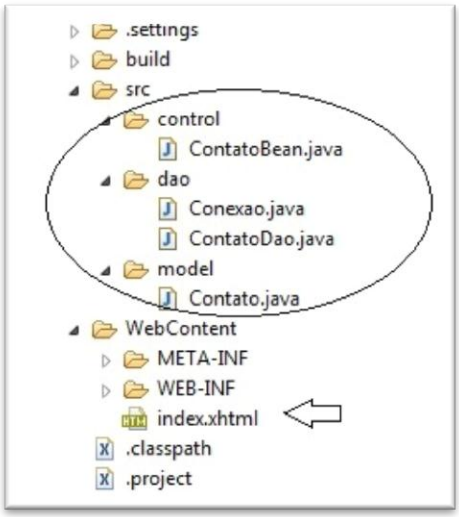

Figura 31 - Estrutura do projeto

Abaixo segue o código fonte das classes:

- Classe ContatoBean

```
package control;

import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

import dao.ContatoDao;
import model.Contato;


@ManagedBean(name= "contatoBean")
@RequestScoped

public class ContatoBean {
	private List<Contato> contatos;

	public ContatoBean(){

	}

	public List<Contato> getContatos() {
		ContatoDao dao = new ContatoDao();
		contatos= dao.listarContatos();
		return contatos;
	}

	public void setContatos(List<Contato> contatos) {
		this.contatos = contatos;
	}
```

```
}
```

- Classe Conexao

```
package dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexao{

	private String driver = "com.mysql.jdbc.Driver";
	private String url = "jdbc:mysql://localhost:3306/agenda";
	private String login = "root";
	private String senha = "";
	private Connection con;

	public Conexao() {

		try {
			Class.forName(driver);
			con = DriverManager.getConnection(url,login, senha);
		}catch (Exception e) {
			e.printStackTrace();
		}
	}

	public Connection getCon() {
		return con;
	}

	public void fecharConexao(){
		try {
			con.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
}
```

- Classe ContatoDao

```
package dao;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import model.Contato;

public class ContatoDao {
	private Conexao con;
	private Statement stmt;
	PreparedStatement pstmt;
```

```
	public ContatoDao(){
		con = new Conexao();
	}

	public ArrayList<Contato> listarContatos(){
		ArrayList<Contato> listaContato = new ArrayList<Contato>();

	  try {
		  stmt=(Statement)con.getCon().createStatement();
		ResultSet rs = stmt.executeQuery("SELECT * FROM contatos");
		while (rs.next()) {
		 Contato c = new Contato();
		 c.setId(rs.getInt("id_contato"));
		 c.setNome(rs.getString("nome"));
		 c.setTelefone(rs.getString("telefone"));
		 listaContato.add(c);
		}
	  } catch (Exception e) {
		e.printStackTrace();
	  } finally {
		con.fecharConexao();
	  }
	  return listaContato;
	}
}
```

- Classe Contato

```
package model;

public class Contato {
	private int id;
	private String nome;
	private String telefone;

	public Contato(){
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public String getTelefone() {
```

```
            return telefone;
      }

      public void setTelefone(String telefone) {
            this.telefone = telefone;
      }
}
```

No diretório WebContent de cada projeto foi criada uma página index.xhtml, o conteúdo de cada página pode ser visualizado nas linhas abaixo.

- Icefaces:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:icecore="http://www.icefaces.org/icefaces/core"
    xmlns:ace="http://www.icefaces.org/icefaces/components"
    xmlns:ice="http://www.icesoft.com/icefaces/component">
    <h:head>
        <title>ICEfaces 3.3.0</title>
        <ice:outputStyle href="./xmlhttp/css/rime/rime.css" />
    </h:head>

    <h:form id="form1">
            <style type="text/css">
               /* Important required because row */
               .ui-datatable-odd {
                   background-color:lightgray !important;
                   background-position-y: -1400px; /* IE7 hack */
               }
         </style>
         <ace:dataTable id="contato" value="#{contatoBean.contatos}"
         var="contato">
           <f:facet name="header">
             Contatos cadastrados
           </f:facet>

           <ace:column id="id" headerText="ID" sortBy="#{contato.id}">
               <h:outputText id="idCell" value="#{contato.id}"/>
           </ace:column>

           <ace:column id="nome" headerText="Nome" sortBy="#{contato.nome}" >
               <h:outputText id="nameCell" value="#{contato.nome}"/>
           </ace:column>

           <ace:column id="telefone" headerText="Telefone"
           sortBy="#{contato.telefone}">
                <h:outputText id="telefoneCell" value="#{contato.telefone}"/>
```

```
          </ace:column>
        </ace:dataTable>
     </h:form>
</html>
```

- Richfaces:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
        xmlns:a4j="http://richfaces.org/a4j"
        xmlns:rich="http://richfaces.org/rich">

<h:head>
      <title>Richfaces 4.3.7</title>
</h:head>

<body>
      <h:outputStylesheet>
      .even-row {
    background-color: #FCFFFE;
    }
    .odd-row {
    background-color: #ECF3FE;
    }
    .active-row {
    background-color: #FFEBDA !important;
    }
     .col1{
     width:500px;
     }
    </h:outputStylesheet>

    <h:form id="form">
        <rich:dataTable value="#{contatoBean.contatos}" var="contato"
         id="table" rowClasses="odd-row, even-row" columnClasses="col1,col1,col1">
            <f:facet name="header">
             Contatos cadastrados
            </f:facet>

            <rich:column>
                <f:facet name="header">
                    <h:outputText value="ID " />
                </f:facet>
                <h:outputText value="#{contato.id}" />
            </rich:column>

            <rich:column>
                <f:facet name="header">
                    <h:outputText value="Nome" />
                </f:facet>
                <h:outputText value="#{contato.nome}" />
            </rich:column>

            <rich:column>
```

```
                <f:facet name="header">
                    <h:outputText value="Telefone" />
                </f:facet>
                <h:outputText value="#{contato.telefone}" />

            </rich:column>
             </rich:dataTable>
    </h:form>
</body>
</html>
```

- Primefaces

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">
<h:head>
<title>Primefaces 5</title>
</h:head>

<body>
      <h:form id="form">
         <p:dataTable id= "tableContato" var="contato"
         value="#{contatoBean.contatos}" >

            <f:facet name="header">
             Contatos cadastrados
            </f:facet>

            <p:column  id="id" headerText="Id Contato" >
             #{contato.id}
             </p:column>

            <p:column id="nome" headerText="Nome">
             #{contato.nome}
             </p:column>

            <p:column id="telefone" headerText="Telefone" >
                #{contato.telefone}
            </p:column>
             </p:dataTable>
    </h:form>
</body>
</html>
```

Seguindo o tutorial início rápido e o guia do usuário do Neoload [12], montamos o cenário de teste com população constante de dez usuários, tempo duração de dois minutos com Google Chrome como browser. A ação analisada no teste foi a listagem de cem registros que deveriam ser exibidos em um dataTable simples.
O resumo estatístico para cada biblioteca pode ser visualizado nas tab.13.

	Icefaces	Primefaces	Richfaces
Total de páginas	190	130	26
Total de acessos	530	230	106
Total de usuários lançados	10	10	2
Total iterações concluídas	40	30	10
Rendimento total (MB)	58.44	14.22	1.24
Total de erros	140	110	16
Total de erros de ação	0	0	0
Média de páginas por segundo	1.6	1.1	0.2
Média de acessos por segundo	4.4	1.9	0.9
Tempo médio de resposta por seg	1.2 s	1.26 s	0.103 s
Tempo médio de resposta por pedido	2.51 s	2.13 s	0.287 s
Vazão média	3.86 Mb/s	0.94 Mb/s	0.08 Mb/s
Taxa de erro	26.4 %	47.8 %	15.1 %
Alertas totais de duração	0 %	0 %	0 %

Tabela 13- Resumo estatístico Neoload

O resumo estatístico exibe os seguintes indicadores globais:

- Total de páginas é o número total de páginas jogadas.
- Total de acessos é o número total de hits (pedidos) jogados.
- Total de usuários lançados é o número total de instâncias de usuário lançadas.
- Total de iterações concluídas - soma do número de vezes em que as ações container são executadas para cada usuário virtual.

- Rendimento total - soma dos tamanhos das respostas a todos os pedidos. (Não inclui upstream).
- Total de erros - é o número total de hits (pedidos) com erros.
- Total de erros de ação - número total de ações lógicas (por exemplo, ações de JavaScript) com erros
- Média de páginas por segundo - número médio de páginas jogadas por segundo.
- Média de acessos por segundo - número médio de visitas (pedidos) jogados por segundo.
- Tempo médio de resposta do pedido - média dos tempos de resposta dos pedidos.
- Tempo médio de resposta da página - média de páginas de resposta do período.
- Vazão média - taxa de transferência média de respostas do servidor (a jusante).
- Taxa de erro - porcentagem de acertos (pedidos) com erros fora do número total de hits.
- Alertas total de duração - porcentagem da duração total de alertas para fora da duração total do ensaio.

Anexo B. Classes e páginas xhtml do projeto Comparação

- Classes do Pacote model

```
package model;

public class Criterio {
	private int id;
	private String descricao;
	private double nota;
	private double peso;

	public Criterio(){

	}

	public Criterio(int id, String descricao, double nota, double peso) {
		this.id=id;
		this.descricao = descricao;
		this.nota = nota;
		this.peso = peso;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getDescricao() {
		return descricao;
	}

	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}

	public double getNota() {
		return nota;
	}

	public void setNota(double nota) {
		this.nota = nota;
	}

	public double getPeso() {
		return peso;
	}

	public void setPeso(double peso) {
		this.peso = peso;
	}
}
```

```java
package model;

import java.util.List;

public class Framework {
	private int id;
	private String nome;
	private double nota;
	private List<Criterio> listaDeCriterios;

	public Framework(){

	}

	public Framework(int id,String nome, double nota, List<Criterio>
listaDeCriterios) {
		this.id = id;
		this.nome=nome;
		this.nota = nota;
		this.listaDeCriterios = listaDeCriterios;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public double getNota() {
		return nota;
	}

	public void setNota(double nota) {
		this.nota = nota;
	}

	public List<Criterio> getListaDeCriterios() {
		return listaDeCriterios;
	}

	public void setListaDeCriterios(List<Criterio> listaDeCriterios) {
		this.listaDeCriterios = listaDeCriterios;
	}
}
```

- Classe do pacote control

```
package control;

@ManagedBean(name = "criterioBean")
@RequestScoped
public class CriterioBean {

	private List<Criterio> criterios = new ArrayList<Criterio>();
	private List<Framework> frameworks;
	private int[] selectedFrameworks;

	public CriterioBean() {
		if(criterios.isEmpty()){
			criterios = new CriterioDao().lista_criterios();
		}
	}

	public void novo(){
		criterios = new CriterioDao().lista_criterios_BD();
		new CriterioDao().atualizaCriterios(criterios);
	}

	public List<Criterio> getCriterios() {
		return criterios;
	}
	public void setCriterios(List<Criterio> criterios) {
		this.criterios = criterios;
	}

	public List<Framework> getFrameworks() {
		frameworks = new CriterioDao().lista_frameworks();
		return frameworks;
	}

	public void setFrameworks(List<Framework> frameworks) {
		this.frameworks = frameworks;
	}

	public int[] getSelectedFrameworks() {
		return selectedFrameworks;
	}

	public void setSelectedFrameworks(int[] selectedFrameworks) {
		this.selectedFrameworks = selectedFrameworks;
	}

	public void onCellEdit(CellEditEvent event) {
		new CriterioDao().atualizaCriterios(criterios);

		Object oldValue = event.getOldValue();
		Object newValue = event.getNewValue();
		if(newValue != null && !newValue.equals(oldValue)) {
			FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_INFO,
"Célula alterada", "Antigo: " + oldValue + ", Novo:" + newValue);
			FacesContext.getCurrentInstance().addMessage(null, msg);
		}
	}
```

```
	public boolean checkSomaPesos(){
		double soma = 0;
		for (int i = 0; i < criterios.size(); i++) {
			soma = (soma+ criterios.get(i).getPeso());
		}
		BigDecimal b = new BigDecimal(soma);
		soma=b.setScale(2,BigDecimal.ROUND_HALF_UP).doubleValue();
		if(soma!=1){
			showMessagePesoInvalido(soma);
			return false;
		}else{
			return true;
		}
	}

	public void analise(){
		boolean b = checkSomaPesos();
		if(b==true){
			//busca as notas de cada framework para cada criterio e guarda
os frameworks em uma lista
			ArrayList<Framework> listaFrameworks = new
ArrayList<Framework>();
			for (int i = 0; i < selectedFrameworks.length; i++) {
				Framework f = new Framework();
				ArrayList<Criterio> lista = new
CriterioDao().buscaNotas(selectedFrameworks[i]);
				f.setId(selectedFrameworks[i]);
				f.setListaDeCriterios(lista);
				listaFrameworks.add(f);
			}

			//inclue os pesos nos criterios de cada framework da lista
			for (int i = 0; i < criterios.size(); i++) {

				for (int j = 0; j < listaFrameworks.size(); j++) {

					for (int k = 0; k <
listaFrameworks.get(j).getListaDeCriterios().size(); k++) {

						if(criterios.get(i).getId() ==
listaFrameworks.get(j).getListaDeCriterios().get(k).getId()){


	listaFrameworks.get(j).getListaDeCriterios().get(k).setDescricao(criterios.g
et(i).getDescricao());

	listaFrameworks.get(j).getListaDeCriterios().get(k).setPeso(criterios.get(i)
.getPeso());

						}
					}
				}
			}
			double nota = 0;
			double soma=0;
			double valor = 0;

	//para cada framework guardar a soma das multiplicações (nota/3 * peso)
```

```
                  for (int i = 0; i < listaFrameworks.size(); i++) {

                        for (int j = 0; j <
listaFrameworks.get(i).getListaDeCriterios().size(); j++) {
                              nota =
listaFrameworks.get(i).getListaDeCriterios().get(j).getNota()/3;


      valor=nota*listaFrameworks.get(i).getListaDeCriterios().get(j).getPeso();

                              soma=soma+valor;
                        }
                        listaFrameworks.get(i).setNota(soma);
                        soma=0;
                  }

                  //salvando o nome em cada objeto framework
                  for (int i = 0; i < frameworks.size(); i++) {
                        for (int j = 0; j < listaFrameworks.size(); j++) {

      if(frameworks.get(i).getId()==listaFrameworks.get(j).getId()){

      listaFrameworks.get(j).setNome(frameworks.get(i).getNome());
                              }
                        }
                  }

                  //exibe a mensagem com o resultado
                  String texto = "";
                  for (int i = 0; i < listaFrameworks.size(); i++) {
                        texto = texto +listaFrameworks.get(i).getNome() + " - "
+ new
BigDecimal(listaFrameworks.get(i).getNota()).setScale(2,BigDecimal.ROUND_HALF_UP).
doubleValue() +"    ";
                  }
                  showMessage(texto);
            }
      }

      public void showMessage(String texto) {
            FacesMessage message = new FacesMessage(FacesMessage.SEVERITY_INFO,
"Resultados: ", texto);
            RequestContext.getCurrentInstance().showMessageInDialog(message);
      }
      public void showMessagePesoInvalido(double valor) {
            FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_INFO,"Mensagem: ", "A soma"
                        + " dos pesos deve ser igual a um, valor atual: " +
valor));

      }
}
```

- Classe do pacote dao

```
public class Conexao{
	private String driver = "com.mysql.jdbc.Driver";
	private String url = "jdbc:mysql://localhost:3306/comparacao";
	private String login = "root";
	private String senha = "";
	private Connection con;

	public Conexao() {
		try {
			Class.forName(driver);
			con = DriverManager.getConnection(url,login, senha);
		}catch (Exception e) {
			e.printStackTrace();
		}
	}

	public Connection getCon() {
		return con;
	}

	public void fecharConexao(){
		try {
			con.close();
		} catch (SQLException e) {
			e.printStackTrace();
	}
	}  }
public class CriterioDao {
	private Conexao con;
	private Statement stmt;

	public CriterioDao(){
		con = new Conexao();
	}

	public ArrayList<Framework> lista_frameworks(){
		ArrayList<Framework> lista = new ArrayList<Framework>();

		try {
			 stmt=(Statement)con.getCon().createStatement();
			ResultSet rs = stmt.executeQuery("SELECT * FROM framework");
			while (rs.next()) {
			 Framework f = new Framework();
			 f.setId(rs.getInt("id_framework"));
			 f.setNome(rs.getString("nome"));
			 lista.add(f);
			}
		} catch (Exception e) {
	  e.printStackTrace();
		} finally {
	  con.fecharConexao();
		}
		return lista;
	}

	public ArrayList<Criterio> lista_criterios(){
```

```
            ArrayList<Criterio> lista = new ArrayList<Criterio>();
            try {
                    stmt=(Statement)con.getCon().createStatement();
                  ResultSet rs = stmt.executeQuery("SELECT * FROM criterio");
                  while (rs.next()) {
                   Criterio c = new Criterio();
                   c.setId(rs.getInt("id_criterio"));
                   c.setDescricao(rs.getString("nome_Criterio"));
                   c.setPeso(rs.getDouble("peso"));
                   lista.add(c);
                  }
            } catch (Exception e) {
        e.printStackTrace();
            } finally {
        con.fecharConexao();
            }
            return lista;
      }

      public ArrayList<Criterio> lista_criterios_BD(){
            ArrayList<Criterio> lista = new ArrayList<Criterio>();

            try {
                    stmt=(Statement)con.getCon().createStatement();
                  ResultSet rs = stmt.executeQuery("SELECT * FROM criterio_bd");
                  while (rs.next()) {
                   Criterio c = new Criterio();
                   c.setId(rs.getInt("id"));
                   c.setDescricao(rs.getString("nome"));
                   c.setPeso(rs.getDouble("peso"));
                   lista.add(c);
                  }
            } catch (Exception e) {
        e.printStackTrace();
            } finally {
        con.fecharConexao();
            }
            return lista;
      }

      //altera a tabela criterio com os dados de citerio_db
      public void atualizaCriterios(List<Criterio> lista){
            String atualiza = ("UPDATE criterio set peso=? WHERE id_criterio=
?");
            try {
                  PreparedStatement pstmt =
con.getCon().prepareStatement(atualiza);
                  for (int i = 0; i < lista.size(); i++) {

                        pstmt.setDouble(1, lista.get(i).getPeso());
                        pstmt.setInt(2,lista.get(i).getId());
                        pstmt.executeUpdate();
                  }
            } catch (Exception e) {
           e.printStackTrace();
       } finally {
           con.fecharConexao();
       }
   }
```

```
	//traz todas as notas de um framework
	public ArrayList<Criterio> buscaNotas(int id_framework){
		ArrayList<Criterio> lista = new ArrayList<Criterio>();

		try {
			stmt=(Statement)con.getCon().createStatement();
		ResultSet rs = stmt.executeQuery("SELECT * FROM nota where
id_framework="+id_framework);
		while (rs.next()) {
		 Criterio c = new Criterio();
		 c.setId(rs.getInt("id_criterio"));
		 c.setNota(rs.getDouble("valor"));
		 lista.add(c);
		}

		} catch (Exception e) {
	  e.printStackTrace();
		} finally {
	  con.fecharConexao();
		}
		return lista;
	}

}
```

- Páginas web

- Comparacao.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
	xmlns:h="http://java.sun.com/jsf/html"
	xmlns:f="http://java.sun.com/jsf/core"
	xmlns:ui="http://java.sun.com/jsf/facelets"
	xmlns:p="http://primefaces.org/ui">

<h:head></h:head>
<body>
	<div align="center">

	<h:form id="form">
		<p:growl id="msgs" showDetail="true"/>
		<p:messages id="messages" showDetail="true" autoUpdate="true"
		closable="true" />

			<br/><br/><br/>

			<p:selectCheckboxMenu id="menu"
			value="#{criterioBean.selectedFrameworks}"
			label="Selecione os Frameworks"
			panelStyle="width:550px">

		<f:selectItems value="#{criterioBean.frameworks}" var = "framework"
		itemLabel="#{framework.nome}" itemValue = "#{framework.id}" />
		</p:selectCheckboxMenu>
```

```
            <br/><br/>

            <p:contextMenu for="criterios" widgetVar="cMenu">
                  <p:menuitem value="Edit Cell" icon="ui-icon-search"
                  onclick="PF('cellCriterios').showCellEditor();return false;"/>
                  <p:menuitem value="Hide Menu" icon="ui-icon-close"
                  onclick="PF('cMenu').hide()"/>
            </p:contextMenu>

            <p:outputPanel id="idPanelPermissao">
                  <p:dataTable id="criterios" var="cr"
                  value="#{criterioBean.criterios}" editable="true"
                  style="width:600px" scrollable="true" scrollHeight="350"
                   editMode="cell" widgetVar="cellCriterios">

                        <p:ajax event="cellEdit"
                        listener="#{criterioBean.onCellEdit}"
                        update=":form:msgs" />

                        <f:facet name="header">
                         Critérios/Pesos
                        </f:facet>

                         <p:column headerText="Id" style="width:17px" >
                              <h:outputText value="#{cr.id}" />
                         </p:column>

                         <p:column headerText="Descrição">
                               <h:outputText value="#{cr.descricao}" />
                         </p:column>

                         <p:column headerText="Peso" style="width:45px" >
                               <p:cellEditor>
                                    <f:facet name="output">
                                          <h:outputText value="#{cr.peso}" />
                                    </f:facet>

                                    <f:facet name="input">
                                           <p:inputText value="#{cr.peso}"/>
                                    </f:facet>

                               </p:cellEditor>
                         </p:column>
                  </p:dataTable>
                  <br/>

            <p:commandButton value="Enviar"
            actionListener="#{criterioBean.analise}"/>

      </p:outputPanel>
   </h:form>
</div>
</body>
</html>
```

- Index.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">

<h:head></h:head>
<body>
      <div align="center">
            <h:form>
                  <p:commandButton value= "Gerar Comparação"
                  action = "comparacao.faces"
                  actionListener="#{criterioBean.novo}"/>
            </h:form>
      </div>
</body>
</html>
```

Referências

[1] ALGAWORKS. Desenvolvimento web com Java Server Faces 2 ª edição, Setembro 2010. Algaworks Softwares e Treinamento.

[2] AMAZON. Último acesso: 30/06/2014. Disponível em: http://www.amazon.com/

[3] ASEL, T. Auswirkung großer Komponentenbäume auf die Performance von JSF Implementierungen; Dezembro de 2012. Último acesso: 30/06/2014. Disponível em : http://www.oio.de/public/java/studie-jsf-mojarra-myfaces-performance-vergleich.htm

[4] BATISTA, M.C.M. Otimização de acesso em um sistema de integração de dados através do uso de caching e materialização de dados. Dissertação de mestrado. Universidade Federal de Pernambuco, Recife. 2003

[5] CAELUM. Java com Testes, JSF, Web Services e Design Patterns.

[6] CARMISINI, A.; VAHLDICK, A. Comparativo entre frameworks JavaServer Faces: Apache Tobago, Primefaces e Richfaces. REAVI - Revista Eletrônica do Alto Vale do Itajaí. Nº02 Dezembro de 2012.

[7] COL, A.; NESELLO, F. Aplicativo web com JSF 2.0 e primefaces para gerenciamento de requisitos de software. Monografia. Universidade Tecnológica Federal do Paraná, Pato Branco. 2013

[8] COSTA, A.; CAMARGO, V. Uma estrutura conceitual para a escolha de frameworks de componentes para desenvolvimento de interfaces ricas. ISSN 2316-2872. T.I.S. São Carlos, v.2, p.26-34, Jan-abr2013.

[9] DEITEL, H. M. ; DEITEL, P . J. Java – Como Programar, 8ª Edição, Editora Pearson 2010. 1114f.

[10] ESTRADA, M. Analisis comparativo entre los frameworks myfaces, icefaces, y richfaces al sistema nutricional de la ESPOCH. Tesis de Grado. Escuela de Ingenieria em Sistemas. 2013.

[11] GEARY, D.; HORSTMANN, C. Core JavaServer Faces 3 edição. ISBN 978-0-13-701289-3. Editora Alta Books. 2012.

[12] GONÇALVES, E. Ajax com JSF1.2 utilizando Jboss Richfaces. Último acesso: 20/05/2014. Último acesso: 30/06/2014. Disponível em: http://www.edsongoncalves.com.br/tag/richfaces/

[13] GOOGLE TRENDS. Último acesso: 30/06/2014. Disponível em: https://www.google.com.br/trends/

[14] GOOGLE. Último acesso: 30/06/2014. Disponível em: https://www.google.com.br/

[15] GORLA, J. P.; FOSCHINI, I. Arquitetura para desenvolvimento web baseado em jsf 2.0 utilizando padrões de projeto. ISSN 231 6-2872 T.I.S. São Carlos, v. 2, n. 3, p. 230-241. Departamento de Computação, Universidade Federal de São Carlos, São Paulo. Set-dez 2013.

[16] ICEFACES - Documentação. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/java/community/documentation.jsf

[17] ICEFACES - Releases Notes. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/wiki/display/ICE/ICEfaces+3.3.0+Release+Notes

[18] ICEFACES - Showcase. Último acesso: 30/06/2014. Disponível em: http://icefaces-showcase.icesoft.org/showcase.jsf

[19] ICEFACES. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/java/home.jsf

[20] ICEFACES Download. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/java/downloads/icefaces-downloads.jsf

[21] ICEFACES Forum. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/JForum/forums/list.page#sthash.1s4DQE4u.dpbs

[22] ICEFACES JIRA. Último acesso: 30/06/2014. Disponível em: http://jira.icesoft.org/browse/ICE/?selectedTab=com.atlassian.jira.jira-projects-plugin:issues-panel

[23] ICEFACES Suporte profissional. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/java/services/icefaces-professional-services.jsf

[24] ICEFACES -Treinamento. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/java/training/icefaces-self-serve-training.jsf

[25] ICEFACES. Adicionando Icefaces em sua aplicação. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/wiki/display/ICE/Adding+ICEfaces+to+Your+Application

[26] ICEFACES. Componentes ECA e Primafaces FAQ. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/java/projects/ICEfaces/ace-primefaces-faq.jsf

[27] ICEFACES. Criando uma aplicação Icefaces com Eclipse. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/wiki/display/ICE/Creating+ICEfaces+Applications+with+Eclipse

[28] ICEFACES. Funcionalidades do Icefaces. Último acesso: 30/06/2014. Disponível em: http://www.icesoft.org/wiki/display/ICE/ICEfaces+Features

[29] JAVASERVER FACES.ORG. Especificações JavaServer Faces. Último acesso: 30/06/2014. Disponível em: http://www.javaserverfaces.net/specification/expert-group

[30] MANN, K. D. Java Server Faces in Action. Manning Publications. ISBN 1-932394-12-5. Ano: 2005.

[31] MARCHIONE, F. Primefaces vs Richfaces vs Icefaces. Artigo web. Ano: 2012. Último acesso: 20/07/2014. Disponível em: http://www.mastertheboss.com/richfaces/primefaces-vs-richfaces-vs-icefaces

[32] MENEZES, R.M. C. JavaServer Faces: Um estudo comparativo entre bibliotecas de componentes. Monografia. Universidade Tiradentes, 2008.

[33] NEOLOAD - Guia do usuário 4.2 . Último acesso: 15/07/2014. Disponível em: http://www.neotys.com/documents/doc/neoload/latest/en/html/#1755.htm

[34] NEOLOAD. Último acesso: 15/07/2014. Disponível em: http://neoload.com/

[35] NÓBREGA, P.B.M. Bibliotecas de Interface Rica no JSF2 - Um comparativo entre Primefaces, richfaces e Icefaces. Revista Java Magazine, Edição 114, p 20-34.

[36] ORACLE. Faces Context. Último acesso: 15/07/2014. Disponível em: http://docs.oracle.com/javaee/7/api/javax/faces/context/FacesContext.html

[37] PINGDOM. Último acesso: 05/06/2014. Disponível em: http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/

[38] PITONAK, Pavol. Comparison of Ajax JSF libraries functionality and interoperability. 96f. Tese. Masaryk University. 2011.

[39] PRIMEFACES Google Code. Último acesso: 15/07/2014. Disponível em: https://code.google.com/p/primefaces/

[40] PRIMEFACES Guia do usuário. Último acesso: 15/07/2014. Disponível em: http://www.primefaces.org/docs/guide/primefaces_user_guide_5_0.pdf

[41] PRIMEFACES Documentação. Último acesso: 15/07/2014. Disponível em: http://www.primefaces.org/documentation

[42] PRIMEFACES Showcase. Último acesso: 15/07/2014. Disponível em: http://www.primefaces.org/showcase/

[43] PRIMEFACES Blog. Último acesso: 15/07/2014. Disponível em: http://blog.primefaces.org/?p=1692

[44] PRIMEFACES Fórum. Último acesso: 15/07/2014. Disponível em: http://forum.primefaces.org/

[45] PRIMEFACES Suporte Profissional. Último acesso: 15/07/2014. Disponível em: http://www.primefaces.org/support

[46] PRIMEFACES. Último acesso: 15/07/2014. Disponível em: http://www.primefaces.org/

[47] RICHFACES - Guia do desenvolvedor. Disponível em: http://docs.jboss.org/richfaces/latest_4_3_X/Developer_Guide/en-US/html/

[48] RICHFACES - Mobile : Último acesso: 15/07/2014. Disponível em : https://community.jboss.org/wiki/GettingStartedWithMobileRichFaces

[49] RICHFACES - Showcase. Último acesso: 15/07/2014. Disponível em: http://showcase.richfaces.org/

[50] RICHFACES Fórum. Último acesso: 15/07/2014. Disponível em: https://community.jboss.org/en/richfaces

[51] RICHFACES JIRA. Último acesso: 15/07/2014. Disponível em: https://issues.jboss.org/browse/RF/?selectedTab=com.atlassian.jira.jira-projects-plugin:issues-panel

[52] RICHFACES Releases Notes. Último acesso: 15/07/2014. Disponível em: http://richfaces.jboss.org/download/archive

[53] RICHFACES. Último acesso: 15/07/2014. Disponível em: http://richfaces.jboss.org/

[54] RODRIGUES, J. As grandes novidades do JSF 2.0. 4 Linux – Free Software Solutions. Cursos, soluções e serviços baseados em software livres e padrões abertos para ambientes de ação crítica. Último acesso: 20/05/2014. Disponível em: http://pt.slideshare.net/4linuxbr/as-grandes-novidades-do-jsf-20-10668720

[55] SOUZA, F. Criando e Configurando um projeto web JSF2 com Primefaces 3 e cdi. Último acesso: 15/07/2014. Disponível em: http://www.devmedia.com.br/criando-e-configurando-um-projeto-web-jsf-2-primefaces-3-e-cdi/25251

[56] TIOBE. Último acesso: 15/07/2014. Disponível em: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[57] VAZ, M.J.; JUNIOR, S.B. Frameworks RIA para JSF - lado a lado: Entrada de dados. Revista Mundo J. Número 42. Ano VIII, 2010.

[58] VAZ, M.J.; JUNIOR, S.B. Frameworks RIA para JSF - lado a lado: Saída de dados Revista Mundo J. Número 44. Ano VIII, 2010.

[59] WIKIPÉDIA. AJAX. Último acesso: 15/07/2014. Disponível em: http://pt.wikipedia.org/wiki/AJAX_(programação)

[60] WIKIPÉDIA. JQUERY. Último acesso: 15/07/2014. Disponível em: http://pt.wikipedia.org/wiki/JQuery

[61] WIKIPÉDIA. Internet Rica. Último acesso: 15/07/2014. Disponível em: http://pt.wikipedia.org/wiki/Internet_rica

[62] YOU TUBE. Último acesso: 15/07/2014. Disponível em: https://www.youtube.com/?hl=pt&gl=BR

[63] MERRIAN, S. Qualitative Research: A Guide to Design and Implementation, Johnssey-Bass, 2ª edition. 2009.